Multiple Session 3D Reconstruction using RGB-D Cameras

Examensarbete utfört i Elektroteknik
vid Tekniska högskolan vid Linköpings universitet
av

Nikolaus Widebäck West

LiTH-ISY-EX--14/4814--SE

Linköping 2014
Multiple Session 3D Reconstruction using RGB-D Cameras

Examensarbete utfört i Elektroteknik vid Volumnatal AB
av

Nikolaus Widebäck West

Handledare: Marcus Wallenberg
ISY, Linköpings universitet
Miroslav Kobetski
Volumnatal AB

Examinator: Klas Nordberg
ISY, Linköpings universitet

Linköping, 8 december 2014
3D-rekonstruktion med RGB-D kamera över multipla sessioner

In this thesis we study the problem of multi-session dense RGB-D SLAM for 3D reconstruction. Multi-session reconstruction can allow users to capture parts of an object that could not easily be captured in one session, due for instance to poor accessibility or user mistakes. We first present a thorough overview of single-session dense RGB-D SLAM and describe the multi-session problem as a loosening of the incremental camera movement and static scene assumptions commonly held in the single-session case. We then implement and evaluate several variations on a system for doing two-session reconstruction as an extension to a single-session dense RGB-D SLAM system.

The extension from one to several sessions is divided into registering separate sessions into a single reference frame, re-optimizing the camera trajectories, and fusing together the data to generate a final 3D model. Registration is done by matching reconstructed models from the separate sessions using one of two adaptations on a 3D object detection pipeline. The registration pipelines are evaluated with many different sub-steps on a challenging dataset and it is found that robust registration can be achieved using the proposed methods on scenes without degenerate shape symmetry. In particular we find that using plane matches between two sessions as constraints for as much as possible of the registration pipeline improves results.

Several different strategies for re-optimizing camera trajectories using data from both sessions are implemented and evaluated. The re-optimization strategies are based on re-tracking the camera poses from all sessions together, and then optionally optimizing over the full problem as represented on a pose-graph. The camera tracking is done by incrementally building and tracking against a TSDF volume, from which a final 3D mesh model is extracted. The whole system is qualitatively evaluated against a realistic dataset for multi-session reconstruction. It is concluded that the overall approach is successful in reconstructing objects from several sessions, but that other fine grained registration methods would be required in order to achieve multi-session reconstructions that are indistinguishable from single-session results in terms of reconstruction quality.

Keywords 3D-Reconstruction, SLAM, RGB-D, 3D-Keypoints, Registration
Abstract

In this thesis we study the problem of multi-session dense RGB-D SLAM for 3D reconstruction. Multi-session reconstruction can allow users to capture parts of an object that could not easily be captured in one session, due for instance to poor accessibility or user mistakes. We first present a thorough overview of single-session dense RGB-D SLAM and describe the multi-session problem as a loosening of the incremental camera movement and static scene assumptions commonly held in the single-session case. We then implement and evaluate several variations on a system for doing two-session reconstruction as an extension to a single-session dense RGB-D SLAM system.

The extension from one to several sessions is divided into registering separate sessions into a single reference frame, re-optimizing the camera trajectories, and fusing together the data to generate a final 3D model. Registration is done by matching reconstructed models from the separate sessions using one of two adaptations on a 3D object detection pipeline. The registration pipelines are evaluated with many different sub-steps on a challenging dataset and it is found that robust registration can be achieved using the proposed methods on scenes without degenerate shape symmetry. In particular we find that using plane matches between two sessions as constraints for as much as possible of the registration pipeline improves results.

Several different strategies for re-optimizing camera trajectories using data from both sessions are implemented and evaluated. The re-optimization strategies are based on re-tracking the camera poses from all sessions together, and then optionally optimizing over the full problem as represented on a pose-graph. The camera tracking is done by incrementally building and tracking against a TSDF volume, from which a final 3D mesh model is extracted. The whole system is qualitatively evaluated against a realistic dataset for multi-session reconstruction. It is concluded that the overall approach is successful in reconstructing objects from several sessions, but that other fine grained registration methods would be required in order to achieve multi-session reconstructions that are indistinguishable from single-session results in terms of reconstruction quality.
Acknowledgments

I would like to thank my two supervisors Marcus Wallenberg and Miroslav Kobetski for their guidance and feedback throughout the whole process. I also wish to thank my examiner Klas Nordberg and everyone at Volumental, in particular Magnus Burenius for thoughtful feedback. Lastly, I would like to thank my friends and loved ones for everything else.
# Contents

1 Introduction ......................................................... 1  
  1.1 Motivation .................................................... 1  
  1.2 Purpose and Background ..................................... 2  
  1.3 Problem Statement ........................................... 3  

2 Theory ........................................................................... 5  
  2.1 Single-Session Dense SLAM for 3D Reconstruction ......... 5  
    2.1.1 Important Definitions and Notation .................. 6  
    2.1.2 Relevant Assumptions ................................... 9  
    2.1.3 Incrementally Building A TSDF From Depth Frames 9  
    2.1.4 Tracking Against An Incrementally Built TSDF .... 10  
    2.1.5 Pose-Graph Optimization and Loop-Closure ......... 10  
  2.2 The Multi-Session Case .......................................... 11  
    2.2.1 Relaxed Assumptions ..................................... 11  
    2.2.2 The Multi-Session Problem ............................. 11  
    2.2.3 Related Work ............................................. 12  
    2.2.4 Useful Algorithms ....................................... 14  
    2.2.5 ISS ...................................................... 14  

3 Method .......................................................................... 19  
  3.1 Implementation .................................................... 19  
    3.1.1 Registration ............................................... 19  
    3.1.2 Pose Re-Optimization .................................. 25  
    3.1.3 Fusion .................................................... 28  
    3.1.4 Repositioned Objects In-Between Scans .............. 28  
  3.2 Evaluation .......................................................... 30  
    3.2.1 Metrics ..................................................... 30  
    3.2.2 TUM RGB-D Dataset .................................... 30  
    3.2.3 Partial Reconstruction Dataset ......................... 32  
    3.2.4 Experiments ............................................... 33  

4 Results .......................................................................... 37  
  4.1 Rough Alignment Broad Search ............................... 37
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2</td>
<td>Rough Alignment Performance Analysis</td>
<td>41</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Aggregate results</td>
<td>41</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Best Results Per Dataset</td>
<td>46</td>
</tr>
<tr>
<td>4.3</td>
<td>Fine Alignment</td>
<td>50</td>
</tr>
<tr>
<td>4.4</td>
<td>Pose Re-Optimization and Fusion</td>
<td>55</td>
</tr>
<tr>
<td>5</td>
<td>Discussion</td>
<td>59</td>
</tr>
<tr>
<td>5.1</td>
<td>Results</td>
<td>59</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Rough Alignment</td>
<td>59</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Fine Alignment</td>
<td>61</td>
</tr>
<tr>
<td>5.1.3</td>
<td>Pose Re-Optimization</td>
<td>61</td>
</tr>
<tr>
<td>5.2</td>
<td>Method</td>
<td>63</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Implementation</td>
<td>63</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Evaluation</td>
<td>63</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Source Criticism</td>
<td>64</td>
</tr>
<tr>
<td>5.3</td>
<td>3D Reconstruction In a Larger Context</td>
<td>65</td>
</tr>
<tr>
<td>6</td>
<td>Conclusions</td>
<td>67</td>
</tr>
<tr>
<td>6.1</td>
<td>Summary</td>
<td>67</td>
</tr>
<tr>
<td>6.2</td>
<td>Registration</td>
<td>67</td>
</tr>
<tr>
<td>6.3</td>
<td>Pose Re-Optimization and Fusion</td>
<td>68</td>
</tr>
<tr>
<td>6.4</td>
<td>Repositioned Objects</td>
<td>68</td>
</tr>
<tr>
<td>6.5</td>
<td>Future Work</td>
<td>68</td>
</tr>
<tr>
<td>A</td>
<td>Dataset Renderings</td>
<td>73</td>
</tr>
<tr>
<td>A.1</td>
<td>TUM RGB-D Dataset</td>
<td>73</td>
</tr>
<tr>
<td>A.2</td>
<td>Partial Reconstruction Dataset</td>
<td>82</td>
</tr>
<tr>
<td>B</td>
<td>Result Renderings</td>
<td>93</td>
</tr>
<tr>
<td>B.1</td>
<td>Foot: Clean</td>
<td>93</td>
</tr>
<tr>
<td>B.2</td>
<td>Foot: Cluttered</td>
<td>96</td>
</tr>
<tr>
<td>B.3</td>
<td>Foot: Duplicates</td>
<td>99</td>
</tr>
<tr>
<td>B.4</td>
<td>Foot: Repositioned</td>
<td>102</td>
</tr>
<tr>
<td>B.5</td>
<td>Bust: Clean</td>
<td>105</td>
</tr>
<tr>
<td>B.6</td>
<td>Bust: Cluttered</td>
<td>108</td>
</tr>
<tr>
<td>B.7</td>
<td>Bust: Duplicates</td>
<td>111</td>
</tr>
<tr>
<td>B.8</td>
<td>Bust: Repositioned</td>
<td>114</td>
</tr>
<tr>
<td>B.9</td>
<td>Chair</td>
<td>117</td>
</tr>
<tr>
<td>B.10</td>
<td>Bag</td>
<td>120</td>
</tr>
</tbody>
</table>

Bibliography | 123 |
In this chapter we begin by presenting the motivation behind the topic of this thesis. We then proceed to present its purpose and background and detail the specific problem statement of the thesis.

1.1 Motivation

To be able to simply and cheaply create accurate 3D models of anything in our environment is a highly sought after ability. It could be an enabling technology for advances in mass customization, architecture, interior design, rapid prototyping, the quantified-self movement [Kelly and Wolf, 2007] and many more exciting areas.

The last few years have seen an explosion in publications relating to Simultaneous Localization And Mapping (SLAM) or 3D reconstruction with a freely moving colour and depth (RGB-D) camera, accompanied by several both commercial and open source initiatives [Stachniss et al., 2014]. This development has been fuelled by the advent of cheap consumer grade RGB-D cameras such as the Microsoft Kinect.

A 3D model of an object is created by moving an RGB-D camera around to capture an RGB-D video of the object from different angles. The video data is then fused together to generate a 3D model by simultaneously optimizing for the shape of the object and the trajectory of the camera. In practice however, it is hard to consistently produce accurate reconstructions using these sensors in realistic and close to real-time settings, due in-part to the precision and noise properties of these cameras. Another difficulty is that it is often hard for a human to know when data from enough angles has been captured so that the full shape can be
As the camera is often operated by a person, the quality of the reconstruction is highly dependent on the actions of the operator. A common approach to alleviate this problem is to present a live preview of the reconstruction to the user to help guide the scanning, e.g. Tanskanen et al. [2013] and Newcombe et al. [2011]. This approach implies increased hardware requirements on the device used to capture the RGB-D video stream, and although real-time feedback may help, it does not guarantee a high-accuracy, artifact-free reconstruction.

To combat this problem, the following use case is envisioned:

1. The user scans a scene or an object using an RGB-D camera, and the reconstruction results in a triangle mesh model.

2. The user is then presented with the model and is able to mark areas on the model that she is not pleased with or that she does not want to retain in the final result.

3. The user then scans the scene or object again, which is reconstructed by fusing the results of the scans according to user preferences. The user could also potentially reposition the object in-between scans to enable capturing of previously occluded parts.

4. The resulting model is again presented and the user can either finish or continue the process from 2.

### 1.2 Purpose and Background

The purpose of the thesis is to investigate how to best enable the use case described in 1.1. The most important part is the ability to fuse models from several different scans, or sessions, in a way that improves the results. We call this multi-session reconstruction in relation to regular or single-session reconstruction. The second most important part, is the ability to create full models of detached objects by moving them in-between scans. The focus of this thesis is these two goals.

The thesis is conducted at the company Volumental [2014], and the functionality is to be investigated as an extension to an already existing 3D reconstruction system that has elements similar to both Newcombe et al. [2011] and Endres et al. [2012]. This system first tracks the camera’s motion relative to the scene, then produces a Truncated Signed Distance Function (TSD) representation of scanned surfaces, and finally converts the model to a triangle mesh representation. Both speed and robustness are important in a final system and the results from each of the steps in the existing reconstruction pipeline can be used for this purpose.
1.3 Problem Statement

To enable fusion of the results from several scans, the scans must first be registered into the same reference frame, starting from unrelated reference frames for each scan. After registration one could either directly fuse the data from the scans, or first jointly re-optimize the combined set of poses. The goal is to investigate firstly how to best register two scans to each other, and secondly how best to merge two registered scans, with and without objects being repositioned as described in section 1.1. In particular we would like to answer the following questions:

1. How should a system be designed to accurately and robustly register two scans from an RGB-D camera to the same reference frame, as a first step in multi-session dense RGB-D SLAM?

2. What subparts should make up such a system, and how do they affect its accuracy and robustness?

3. How do we best extend an existing system for reconstruction to the multi-session case, given that the scans have been brought to the same reference frame?

4. How do the answers to the above stated questions change if objects are allowed to move in between sessions?
In this chapter we present the problem of multi-session dense SLAM and give an overview of the related theory. In order to do so, we start by giving an overview of the single-session problem and cover important definitions, notation, and assumptions that have implications for solutions to the multi-session case. This is followed by a description of the multi-session problem and how the main assumptions there differ to those presented for the single-session case. The chapter ends with a thorough literature review and a description of some important algorithms that can be used in solving multi-session dense RGB-D SLAM.

2.1 Single-Session Dense SLAM for 3D Reconstruction

The problem of Simultaneous Localization And Mapping (SLAM) was originally formulated in the robotics community as part of the long term goal of creating autonomous robots [Durrant-Whyte and Bailey, 2006]. In the context of 3D-reconstruction with an RGB-D camera, dense SLAM is the approach to building a dense 3D-model by simultaneously solving for the camera poses and dense shape of which the RGB-D frames are noisy images. This is often done by sequentially matching each frame to either the previous frame or an accumulated map, as done in Newcombe et al. [2011]. All sequential matching approaches result in drift in tracking due to error accumulation over time, although less so in the frame-to-map case. In the frame-to-frame matching case, tracking drift is commonly alleviated by running a global optimization over the camera poses, e.g. [Sturm et al., 2012] and [Kerl et al., 2013], where the results from the frame-to-frame tracking are used as initialization. An illustration of the SLAM problem
Figure 2.1: Single session SLAM with a freely moving camera. The camera observes the scene from different vantage points while travelling along its 3D-trajectory.

can be seen in figure 2.1.

2.1.1 Important Definitions and Notation

2.1 Definition (Camera Pose). The pose of an object is a description of its position and orientation. Thus in 3D-space, the camera pose describes the position and direction of the camera in some coordinate system. Formally we represent the camera pose as a transformation $T : \mathbb{R}^3 \mapsto \mathbb{R}^3$, mapping a point $p_c \in \mathbb{R}^3$ in the coordinate system of the camera, to a point $p_g \in \mathbb{R}^3$ in the global coordinate system by

$$\begin{pmatrix} p_g \\ 1 \end{pmatrix} = T_c \begin{pmatrix} p_c \\ 1 \end{pmatrix} = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} p_c \\ 1 \end{pmatrix},$$

which we sometimes write as

$$p_g = T_c p_c = Rp_c + t,$$

where $R \in SO(3)$, the Special Orthogonal Group, is a rotation and $t \in \mathbb{R}^3$ is a translation. The set of such transformations $T$ is called the Special Euclidean Group $\mathbb{SE}(3)$. Further, let $T_{i\leftarrow j} = T_{j}^{-1}T_{i}$ be the relative pose between camera $i$ and $j$ such that $T_{i\leftarrow j} p_j = p_i$. Depending on the context, $i$ and $j$ may also represent different time indices for the same camera. For clarity we may also write $T_c$ as $T_{g\leftarrow c}$, which may be thought of as the relative pose between the camera and the global reference frame.
2.1 Single-Session Dense SLAM for 3D Reconstruction

![Figure 2.2: Pose-Graph SLAM](image)

The nodes of the graph are given by the poses $T_1, T_2, \ldots, T_N$ and the edges are given by observed matchings, each described by a relative pose parameter vector $z_{i,j}$ and an information matrix $\Omega_{i,j}$. Solid lines describe edges from sequential matching and dashed from loop closing.

2.2 Definition (Minimal Parametrization of $SE(3)$). Let $T \in SE(3)$ be a transformation with rotation component $R \in SO(3)$, and translation component $t \in \mathbb{R}^3$. Let $z \in \mathbb{R}^6$ be its minimal parametrization. In this thesis we let

$$z = \begin{pmatrix} w_1 & w_2 & w_3 & t_x & t_y & t_z \end{pmatrix}^T = \begin{pmatrix} w \ t \end{pmatrix},$$

be defined such that

$$R = exp \left( \begin{bmatrix} w \end{bmatrix} \times \right) = \exp \left( \begin{bmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w_1 \\ -w_2 & w_1 & 0 \end{bmatrix} \right),$$

where $\exp$ is the matrix exponential function.

2.3 Definition (Pose-Graph). A pose-graph is the set of nodes $T_{1:N}$ and edges $E$. The nodes, $T_{1:N} = [T_1, T_2, \ldots, T_N]$, are the set of camera poses sampled at times 1 through $N$. An edge, $(i, j) \in E$, represents a constraint between the poses $T_i$ and $T_j$, which arises from a (virtual) measurement of the relative pose $T_i \leftarrow T_j$. The measurement consists of a parameter vector, $z_{i,j}$, which describes the relative pose, and a corresponding information matrix $\Omega_{i,j}$, with $z_{i,j} \in \mathbb{R}^6$ and $\Omega_{i,j} \in \mathbb{R}^{6 \times 6}$.

The pose-graph is used to represent the results from sequential tracking in a way that allows for doing non-sequential estimation of a set of camera poses by solving for the poses that best fit the constraints of the pose-graph. This is called pose-graph optimization and is explained in section 2.1.5. A virtual measurement, $z_{i,j}$ and $\Omega_{i,j}$, is the result of matching the scenes seen by the camera at times $i$ and $j$, for instance using one of the methods described in section 2.1.4. A visualization of a simple pose-graph can be seen in figure 2.2.
Fig. 2.3: Signed distance function

This figure shows a 2D example of an SDF represented on a regular grid. Each entry in the array stores the shortest distance to the curve from the center of the corresponding grid square.

2.4 Definition (SDF). Given a solid shape $A \subset \mathbb{R}^3$, with boundary surface $S = \partial A$, the signed distance function (SDF) to shape $A$ is given by $d(A, p) : \mathbb{R}^3 \mapsto \mathbb{R}$ such that,

$$d(A, p) = \begin{cases} 
0 & \text{if } p \in S, \\
-\min_{s \in S} \|s - p\| & \text{if } p \text{ in } A, \\
\min_{s \in S} \|s - p\| & \text{if } p \text{ in } A^C.
\end{cases}$$

In words, the signed distance between the point $p$ and the shape $A$, is the shortest distance between $p$ and the surface of $A$, with a negative sign if $p$ is inside $A$.

The implicit SDF representation of surface geometry is widely used in computer graphics, including for mesh generation [Persson, 2004], accelerated ray tracing [Hart, 1996], and robust point cloud registration [Fitzgibbon, 2003]. A visualization of an SDF represented on a grid can be seen in figure 2.3.

2.5 Definition (TSDF). The truncated signed distance function (TSDF) is the distance-truncated version of the SDF. Here however, we will take the TSDF to mean the approximation of this function on a discretized grid, such as a voxel grid or octree, as often used for dense SLAM with non-penetrating cameras. With
a truncation distance of $\mu$, we let the TSDF $d_\mu(A, p)$ be given by

$$
d_\mu(A, p) = \begin{cases} 
\mu & \text{if } d(A, p) > \mu, \\
0 & \text{if } d(A, p) < -\mu, \\
d(A, p) & \text{otherwise.}
\end{cases}
$$

As is explained in section 2.1.3, a TSDF is often created from the measurement of a surface from a single view at a time. The zero value for negative distances below $-\mu$, is therefore often taken to mean; ignore this voxel, as it is impossible to know what is behind a surface from a single view. The use of truncation also alleviates other problems due to that a SDF calculated from a single view of a shape can only be counted on to be accurate close to the measured surface.

### 2.1.2 Relevant Assumptions

2.6 Assumption (Incremental Camera Movement). The RGB-D frames are assumed to be sampled sequentially and with a high enough frequency, from a single camera travelling along a continuous 3D-trajectory without large variations in either position or rotation, such that successive camera poses can be considered close.

2.7 Assumption (Static Scene). The object or scene being scanned is, except for any effects due to moderate lighting change, static throughout the duration of the RGB-D video capture.

### 2.1.3 Incrementally Building A TSDF From Depth Frames

Newcombe et al. [2011], Bylow et al. [2013], and Trifonov [2013] among others, use a TSDF as geometry representation when building an incremental map during camera tracking. When tracking is done, a polygon-mesh representation can be extracted directly from the TSDF using an algorithm such as marching cubes [Lorensen and Cline, 1987] or dual contouring [Ju et al., 2002].

Given an estimated camera pose, an approximation of the TSDF is calculated for the sampled surface belonging to that pose. Instead of calculating the true distance from each voxel in the TSDF volume to the surface, the projective distance is used, where the projection is given by a pinhole camera model and the accompanying camera pose. The projective distance is then calculated as the distance, along the viewing ray, between the measured surface and the voxel centre.

The final TSDF is then the weighted average of the TSDF approximations from each frame, where per-voxel weights can be exploited to take things such as noise models, pose uncertainty, etc. into account. The weights are stored along with the distance values such that the TSDF volume can be built incrementally using a running average computation.
2.1.4 Tracking Against An Incrementally Built TSDF

Given a new frame $i$, the goal is to find the camera pose $T_i$ such that the observed surface, $S_i$, is brought into alignment with the surface $S_{0:i-1}$, described by the TSDF incrementally built from frames $T_{0:i-1}$.

Newcombe et al. [2011] register a new frame by first generating a virtual depth frame by ray-tracing the TSDF from the previous camera pose $T_{i-1}$, and then choosing $T_i = T_{i-1}T_{i-1 \rightarrow i}$, where $T_{i-1 \rightarrow i}$ is found to give the best alignment between frame $i$ and the virtual depth frame. $T_{i-1 \rightarrow i}$ is found by applying the Iterative Closest Point (ICP) algorithm for finer and finer resolutions of the depth frames with the identity transformation as the first guess.

Instead of registering the new frame to a virtual frame, Bylow et al. [2013] find $T_i$ by directly registering the frame point cloud $X_i$ to the TSDF. This is done by using a Gauss-Newton optimization scheme to minimize

$$\sum_{x \in X_i} d(T_ix, S_{0:i-1})^2,$$

over $T_i$, using $T_{i-1}$ as the starting guess. It is claimed in [Bylow et al., 2013] that this method gives better tracking performance than the method presented in [Newcombe et al., 2011].

2.1.5 Pose-Graph Optimization and Loop-Closure

The SLAM problem is often represented on a graph, where the most common graph-representation for dense reconstruction is a Pose-Graph [Grisetti et al., 2010], described in definition 2.3. In Pose-Graph SLAM, we try to solve for the set of camera poses $T_{1:N}$ that best fit the constraints of the SLAM problem as represented on a pose-graph. This is done in order to decrease pose tracking drift from sequential tracking.

The pose-graph is initialized with the results from sequential matching, where a virtual measurement $z_{i,j}$ is the estimated matching parameters between frame $i$ and $j$. The information matrix of the observation, $\Omega_{i,j} = \Sigma_{z_{i,j}}^{-1}$, is the inverse covariance matrix of the objective function of the matching.

We define $\hat{z}_{i,j}$ to be the vector of parameters describing the relative pose $T_i^{-1}T_j$, and choose $T_{1:N}^*$ as:

$$T_{1:N}^* = \arg \min_{T_{1:N}} F(T_{1:N})$$

$$F(T_{1:N}) = \sum_{(i,j) \in E} (\hat{z}_{i,j} - z_{i,j})^T \Omega_{i,j}(\hat{z}_{i,j} - z_{i,j})$$  \hspace{1cm} (2.1)

Both $z_{i,j}$ and $\hat{z}_{i,j}$ are minimal parametrizations (see definition 2.2) of relative pose transformations, which lie in $\mathbb{SE}(3)$. Among other things, avoiding redundant parameters allows for $\Omega_{i,j}$ to be calculated as the inverse of the covariance matrix of the objective function of the frame-to-frame matching.
The observations $z_{i,j}$, where $j = i + 1$, are the estimated relative poses from the sequential matching stage, and if no other observations are used the pose-graph optimization will trivially return the initial set of poses. Any non sequential edges are called loop-closures and the process of adding such observations to the pose-graph is called loop-closure detection.

2.2 The Multi-Session Case

In this section we formulate the multi-session dense RGB-D SLAM problem and present how two important assumptions from the single-session case are modified. In particular, we formulate the multi-session problem as an extension to single-session dense RGB-D SLAM, which we in this context view as a solved problem. We also give an overview of work related to the multi-session problem and list and describe some important algorithms.

2.2.1 Relaxed Assumptions

Multi-session dense RGB-D SLAM differs from the single-session variation primarily through the relaxation of the incremental camera movement assumption. Insofar as this assumption holds in the single-session case, it also holds within each session in the multi-session case. For sessions $A$ and $B$, we thus have two sets of poses $T_{0:N}^A$ and $T_{0:M}^B$, both of which conform to the incremental camera movement assumption internally, but for which we can make no assumptions of closeness between any particular pair of poses from session $A$ and $B$. We do however need some degree of overlap between the two sets of frames for the problem to be solvable, which leads to the following modified assumption.

2.8 Assumption (Session-Wise Incremental Camera Movement). Camera trajectories belonging to each of the SLAM sessions adhere to assumption 2.6. It is further assumed that the reconstructed dense models from each of the involved sessions overlap enough, for matching between models to be feasible.

The static scene assumption is made for single-session SLAM because it greatly simplifies the task of reconstructing the scene. However, the multi-session case allows for slightly breaking this assumption without much added complexity. In a two-session scenario, one could allow a scanned object to undergo rigid-body motion relative the rest of the scene in-between scans. If coupled with some user provided meta-data, e.g. a bounding cuboid of the moved object in one of the scans, the object could be reconstructed in a similar way to as when left static.

2.2.2 The Multi-Session Problem

Given an acceptable solution to dense single-session RGB-D SLAM, solving the multi-session problem reduces to bringing all scans to the same reference frame, potentially re-optimizing the poses, and fusing the data from all the scans. In the context of working single-session dense reconstruction, we divide the multi-session problem into the following sub-problems:
**Registration** The poses belonging to each of the scans must be brought into a common reference frame, or global coordinate system. For scans \( A \) and \( B \), with resulting point clouds \( \mathcal{X}_A \) and \( \mathcal{X}_B \), this can be stated as finding the rigid transformation \( \mathbf{T}_{B,A} : \mathbb{R}^3 \mapsto \mathbb{R}^3 \) such that when applied to each point \( p_A \in A \) by \( \tilde{p}_A = \mathbf{T}_{B,A}p_A \) the transformed point cloud \( \tilde{\mathcal{X}}_A \) is brought into alignment with \( \mathcal{X}_B \).

**Pose Re-optimization** (Optional) Jointly optimize the poses using all the collected data.

**Fusion** The data from all of the scans should be used to create the final reconstruction.

If the static camera assumption is broken as described in 2.2.1, this must also be handled in the steps above. With the formulation of the multi-session problem as taking the results from a single-session SLAM system as input, artefacts and errors from that system must also be taken into account.

### 2.2.3 Related Work

The Iterative Closest Point (ICP) algorithm [Chen and Medioni, 1992] and its many variations are often considered the gold standard of 3D registration algorithms [Castellani and Bartoli, 2012]. ICP is however highly susceptible to getting stuck in local minima and therefore requires a good initialization in order to converge to the global optimum. This is commonly done by first matching a sparse set of keypoints from each model to find the first initial transformation. The method GO-ICP [Jiaolong Yang and Jia, 2013] promises a provably global optimal registration of 3D models but reports speeds that are far too slow for near real-time applications. Rusinkiewicz and Levoy [2001] also show that using normal-space sampling to sub-sample point clouds before registration, gives the best convergence properties for models containing sharp details. In [Bouaziz et al., 2013] ICP with the use of the \( L_p \) norm for \( p < 2 \) is used to gain higher robustness to outliers and noise.

The 3D registration problem, particularly that of the initial transformation estimation, can be equivalently posed as loop closure detection or camera pose re-localization in the context of SLAM. In Williams et al. [2009] loop closure detection techniques for monocular SLAM are classified into the categories Map-to-Map, Image-to-Image, and Image-to-Map. Glocker et al. [2013] classifies online real-time performance capable re-localization methods into either landmark-based approaches or image-based approaches. This classification is made from the perspective of 3D reconstruction and augmented-reality systems.

The method presented in Glocker et al. [2013] falls into the category of image-based, Image-to-Image approaches, and uses a random forests based approach to create fast compact codes of incoming RGB-D frames that are matched against a database of previously seen frames to relocate the camera pose. This method is used in concert with KinectFusion described in Newcombe et al. [2011], which does not use any keypoints during tracking. Although the method is reportedly
both fast and quite simple, because the re-localization is based on matching to previously seen frames, non-overlapping sets of poses that still point to the same object are unlikely to be handled well.

All the methods discussed in Williams et al. [2009] are either pure landmark-based or hybrid methods. A common approach is to use some variation of bag-of-words to speed up matching between detected keypoints when doing loop closure detection. Both Paul and Newman [2010] and Cadena et al. [2012] combine the bag-of-words method with probabilistic models (Random Graphs and Conditional Random Fields respectively) to capture geometric as well as co-appearance information between feature words. Another popular approach is to employ geometric checking followed by RANSAC [Fischler and Bolles, 1981] to do the final matching [Martínez-Carranza and Mayol-Cuevas, 2013].

Williams et al. [2009] conclude that Map-to-Map methods are not robust enough to be used in the monocular SLAM context using sparse keypoint based mapping. The Map-to-Map approach is most similar to the 3D model registration problem as approached in the graphics community [Gelfand et al., 2005, Petricek, 2012, Tam et al., 2013]. In these settings, initialization is usually done by matching 3D keypoints that are detected and described directly on a polygon-mesh or point cloud representation of the objects [Bronstein et al., 2012].

The 3D-keypoint descriptors range from having a complete local reference frame described by position, rotation, and scale, to being descriptors with regard only to a position and a scale. Some descriptors, e.g. NARF [Steder et al., 2010], also encode viewpoint information and are therefore particularly suited to Image-to-Map or Image-to-Image approaches. A thorough evaluation of 3D-keypoint detectors can be found in Tombari et al. [2013]. In Salti et al. [2012] 3D detector/descriptor pairs are evaluated together and it is determined that the best such pair over all is ISS/SHOT [Zhong, 2009, Tombari et al., 2010]. Rusu [2009] presents evidence for the prevalence of planes in indoor environments, and Eugen Ichim [2013] takes this further and uses planes as features in a pose-graph SLAM environment.

In [Aldoma et al., 2012] a pipeline for 3D-object detection in point clouds is presented that makes use of both sparse 3D-keypoint matching as well as global 3D-shape descriptors. Sparse keypoints are matched against a database of objects to create correspondences, which are then clustered to create detection hypotheses. In parallel, the point cloud is segmented, and global shape descriptors are computed and matched for each of the segments to create another set of hypotheses. Hypotheses are refined with ICP and then verified in a hypothesis verification framework. Rusu et al. [2009] uses the Huber error metric on closest point correspondences to weight rough alignment hypotheses as part of the algorithm SAC-IA.

Levihn et al. [2013] and Kim et al. [2013] present methods for detecting 3D objects from a database in range scans, and make use of the detected models to improve the resulting scan. Both methods use viewpoint dependent 3D descriptors
and therefore virtually scan the models in their databases from different angles, for use when detecting objects from incoming scans. Dame et al. [2013] uses a known object shape as a regularizer to a dense monocular SLAM algorithm. The object is first detected and then used to improve the robustness of the camera tracking and reconstruction. A method for fusing together a SDF representation of the known shape with the TSDF volume in the reconstruction is also presented.

The properties of the TSDF shape representation with respect to 3D reconstruction and object detection is explored in Ricoa Canelhas [2012], and a more mathematical treatment of the SDF is given in Mayost [2014] and Goldman [2005]. The LM-ICP [Fitzgibbon, 2003] algorithm for shape registration includes a pre-processing step where the destination shape is converted to a SDF representation to speed up computation. Bylow et al. [2013] uses a Gauss-Newton based approach to efficiently register a depth image to a TSDF volume. Canelhas et al. [2013] showed that registering several depth frames into a TSDF representation to remove noise before detecting and describing 3D-keypoints, greatly improves performance compared to when descriptors are computed on single depth frames.

2.2.4 Useful Algorithms

Here we list and describe a few algorithms that can be important in solving the multi-session problem.

2.2.5 ISS

In [Zhong, 2009] present a method for 3D-keypoint detection and description in point clouds named Intrinsic Shape Signatures (ISS). Although the descriptor has since been outperformed, the detector has been found to be the best performing fixed-scale detector in several surveys [Tombari et al., 2013, Salti et al., 2012].

Each point is first given a weight, $w_i$, which equals the reciprocal of the number of points that lie within the radius $r_{density}$ from the point. The eigenvalues $\lambda_1, \lambda_2, \lambda_3$ in decreasing magnitude, of the weighted covariance matrix $\Sigma(p_i)$ are then analysed, where

$$\Sigma(p_i) = \frac{\sum_{\|p_i - p_j\| < r_{frame}} w_i (p_i - p_j)(p_i - p_j)^T}{\sum_{\|p_i - p_j\| < r_{frame}} w_i}.$$ 

The saliency of a point is defined as $\lambda_3$, and keypoints are chosen by applying non-maximum suppression over saliency after pruning points that do not full-fill:

$$\frac{\lambda_2}{\lambda_1} < \gamma_{21} \text{ and } \frac{\lambda_3}{\lambda_2} < \gamma_{32}.$$ 

This pruning step removes points where the surrounding shape has strong directionally dependent structure, as would be the case if it was located on an edge or
2.2 The Multi-Session Case

FPFH

Rusu et al. [2009] present a method for 3D point description called Fast Point Feature Histograms (FPFH), that is based on creating histograms of point-pair comparisons within a region of the described point. For each point-pair, with center point $p_s$ and comparison point $p_t$, a comparison consists of calculating the three numbers

$$\alpha = \arccos(v \cdot n_t)$$

$$\phi = \arccos(u \cdot \frac{(p_t - p_s)}{\|p_t - p_s\|_2})$$

$$\theta = \arctan(w \cdot n_t, u \cdot n_t),$$

where $n_t$ is the normal at $p_t$, $u = n_s$ is the normal at $p_s$, $v = n_s \times (p_t - p_s)$, and $w = u \times v$.

The histograms of $\alpha$, $\phi$, and $\theta$ between the centre point $p_s$ and all other points in a spherical neighbourhood are concatenated to create a Simple Point Feature Histogram SPFH($p_s$). Finally, the FPFH for a point is calculated as the weighted average of all SPFH within its support radius.

SHOT

Signatures of Histograms of Orientations (SHOT) is a 3D-keypoint descriptor presented by Tombari et al. [2010], which is inspired by the popular 2D-keypoint descriptor SIFT. The support sphere of the feature is partitioned into a spherical grid. For each grid volume, the cosine of the angle between the normal of the feature point and each of the normals at the points in the volume, are binned into a histogram. When binning, quadrilinear interpolation is used, both with neighbouring bins in the local histogram and with bins of the same index in neighbouring volumes, to smooth out quantization effects. The histograms are then concatenated and the full histogram is normalized to add robustness to point density variations.

Geometric Correspondence Grouping

Geometric Consistency Grouping (GCG) is a method for clustering matched point-correspondences between point clouds. Let $A$ and $B$ be two sets of points of the same size, where $p_{i,A}$ and $p_{j,B}$ from point sets $A$ and $B$ respectively are matched correspondences. GCG then clusters correspondences such that a pair of correspondences $(i, j)$ in the same cluster adheres to

$$\left\|p_{i,A} - p_{j,A}\right\| - \left\|p_{i,B} - p_{j,B}\right\| < \epsilon.$$

Finally, RANSAC based outlier rejection is applied to the clusters.
3D Hough Voting for Recognition

In [Tombari and Di Stefano, 2012], a method for clustering matched correspondences that makes use of repeatable local reference frames of feature points is presented. Let $A$, $B$, $p_{i,A}$, and $p_{i,B}$ be given as for GCG. Each point $p_{i,A} \in A$ is first associated with $c_{i,A}$, which is the centroid of the point cloud from which the keypoints $A$ are taken, expressed in the local reference frame of $p_{i,A}$. Each correspondence $(p_{i,A}, p_{i,B})$ then casts a vote in 3D-Hough space for the position of $c_{A}$ in the global reference frame of $B$, by using the position and local reference frame of $p_{i,B}$. The local maximums of the hough space are then taken as clusters, and finally outliers are removed from the clusters using RANSAC.

RANSAC

RANdom SAmple Consensus (RANSAC) [Fischler and Bolles, 1981] is a widely used method for robust model fitting and outlier rejection. In the context of point cloud registration with correspondence sets $A$ and $B$, the model that is sought is the transformation $T : \mathbb{R}^3 \mapsto \mathbb{R}^3$ that maps points in $A$ to their correspondences in $B$. The algorithm consists of repeating the following until convergence:

1. Randomly sample a small subset of correspondences.
2. Using the sampled subset, estimate the model $T$.
3. Label the correspondences for which $\|T p_{i,A} - p_{i,B}\| < \epsilon$ as inliers.
4. If enough points are labelled as inliers, the algorithm has converged.

We then optionally re-estimate $T$ using all of the inliers. If RANSAC is used for outlier rejection we then also discard those correspondences not labelled as inliers.

ICP

Iterative Closest Point (ICP) is an algorithm for fine-grained rigid-registration of roughly aligned geometrical models, that has seen wide usage in geometry processing and exists in many variations [Rusinkiewicz and Levoy, 2001]. Given two point clouds $\mathcal{X}$ and $\mathcal{Y}$, the goal is to find the rigid transformation $T : \mathbb{R}^3 \mapsto \mathbb{R}^3$ such that the cost function

$$F(\mathcal{X}, \mathcal{Y}, T) = \sum_{(i,j) \in C} e(T x_i, y_j),$$

is minimized. $e : \mathbb{R}^3 \times \mathbb{R}^3 \mapsto \mathbb{R}$ is the point wise error function and $C$ is the set of true correspondences, which is unknown. ICP solves this problem by iterating between estimating $C$ and $T$. $C$ is usually estimated by choosing for each point in $T \mathcal{X}$, the closest point in $\mathcal{Y}$, and then using heuristics like a maximum threshold on point-to-point distance to remove bad correspondences. Common choices for
are

\[ e(x, y) = \|x - y\|^2_2 \quad \text{and} \]
\[ e(x, y) = ((x - y) \cdot n_y)^2, \]

referred to as the point-to-point and point-to-plane error metrics respectively. \( n_y \) is here the normal at the point \( y \).
In this chapter we describe the implementation and evaluation of a system for doing multi-session dense reconstruction. Both the implementation and evaluation sections begin by describing the registration step and then describe the pose re-optimization and fusion steps.

3.1 Implementation

In this section we describe the implementation of a system for doing multi-session dense reconstruction on top of a system capable of doing single-session dense reconstruction. Although these methods could be applied to many sets of scans and partial-reconstructions, the specific implementation for this thesis is focused on the merging of two scans. The full system is divided into three subsystems: registration, pose re-optimization, and fusion, which are described below. Finally, the modifications to the system allowing it to handle objects repositioned between scans is presented.

All the application code is written in C++. The implementations make heavy use of the open source library Point Cloud Library (pcl) [Rusu and Cousins, 2011], to which much time was spent getting familiar to. Python, together with parts of the SciPy [Jones et al., 2001–] stack are used for evaluation and as glue code where needed.

3.1.1 Registration

The registration subsystem takes two point clouds with normals as input and outputs the rigid transformation $T : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ that brings the first point cloud $\mathcal{X}$, into alignment with the second point cloud $\mathcal{Y}$. The point clouds can be con-
structed from reconstructed mesh models, where the points are the vertices and the normals are calculated as the area weighted average of the surrounding face normals. We begin with an overview of the whole registration pipeline in two variations, and then list and explain their exchangeable components. Throughout the registration module, all scale parameters are specified relative to the TSDF voxel resolution used when reconstructing the models.

**Pipeline Overview**

The implemented registration pipeline comes in two slightly different variations, which we call the *late-* and *early-integration* pipelines. Both variations are based on the pipeline for 3D-object recognition in point clouds from depth images presented in [Aldoma et al., 2012], and the *late-early* division is a result of the work done on this thesis. Overviews of the *late-* and *early-integration* variations can be seen in figure 3.1 and figure 3.2 respectively.

**Late-Integration Pipeline** The *late-integration* pipeline is essentially an adaptation of the pipeline presented in [Aldoma et al., 2012], which is developed for object detection in depth image point clouds. There are some different properties inherent in the problem of finding a noise-free, relatively small object in a noisy scene, and the problem of matching two large point clouds with potential reconstruction defects. However, the structure of the pipeline is basically the same. The main structural difference is that hypothesis verification is performed before fine alignment, because of the relatively high cost of aligning two large clouds using ICP. Another difference is that planes are used directly as large shape descriptors instead of first doing segmentation and then applying a complex global shape descriptor.

The pipeline is divided into the rough and fine alignment parts. In the rough alignment part, the strategy is to generate several hypotheses, choose the best seeming hypothesis, and pass it on to the fine alignment part, which consists of registering the two point clouds using ICP. Hypotheses are generated both by detecting and matching sparse keypoints, as well as matching detected planes.

**Early-Integration Pipeline** The *early-integration* pipeline diverges further from the pipeline presented in [Aldoma et al., 2012] than the *late-integration* pipeline. Instead of using plane matches to generate hypotheses to be pruned in the final stage, the hypothesis verification module is used to select a single good plane match, which is used to constrain the matching of sparse keypoints before hypothesis generation. This pipeline variation therefore requires the presence of a commonly visible plane in both point clouds.
Figure 3.1: Late-Integration Registration Pipeline

The figure above depicts an overview of the late-integration point cloud registration pipeline, with the arrows showing the flow of the data between different pipeline steps. Hypotheses transformations are generated both by sparse keypoint based matching between the point clouds, and from matching planes detected in the point clouds. The best hypothesis is then chosen and passed on to the fine alignment step.
**Figure 3.2: Early-Integration Registration Pipeline**

The figure above depicts an overview of the early-integration point cloud registration pipeline, with the arrows showing the flow of the data between different pipeline steps. Planes from both point clouds are matched to each other. The best plane match is selected by the hypothesis verification step resulting in extra geometric constraints that are used when matching sparse keypoints.
Below we list and describe the exchangeable components of the pipelines.

Filter

**Plane Filter**  RANSAC is used to extract dominant planes from a point cloud. All detected planes that contain at least $\tau\%$ of the point cloud are considered dominant. The ground plane is determined as the plane that is the furthest down along the z-axis (the input point clouds are such that the z-axis is roughly aligned with the real world up-direction). The planes, given by the sub-sampled point-sets and plane parameters, are passed on to the plane matching step. The ground plane and all points below it are then filtered out of the point cloud. This is meant to reduce matching errors due to matching noisy parts of a flat surface, and is primarily motivated by the use case of a person scanning the same object repeatedly on a desk or on the floor.

**Pass Through**  For comparison to the plane filter, this variation only sends an empty set of planes and an unchanged point cloud to the following steps.

Plane Matching

The **plane matching** step matches all detected planes from cloud $\mathcal{X}$ to the detected planes from cloud $\mathcal{Y}$. Each pair of planes gives an hypothesis transformation $T_h$. The rotational component of $T_h$ is first estimated by finding the rotation that brings the first normal in line with the second, and the translation component $t$ as the difference between the two planes’ centroids. The final hypothesis is then found by applying ICP with a point-to-point error metric, to the point-sets of the two planes, with the estimated rotation and translation as starting guesses.

Keypoint Detection

**Uniform Sampling**  The point cloud is first embedded in a voxel-grid with voxel side-lengths $r_{density}$. The keypoints are then chosen as the average of each of the point-sets belonging to a voxel.

**ISS**  Keypoints are detected in the point cloud using the ISS algorithm, described in section 2.2.5.

Sparse Matching

In all cases, the matching between sets of features to generate a set of correspondences is done by an approximate nearest neighbour computation using the open source library FLANN [Muja and Lowe, 2009]. The point descriptors are either calculated with the FPFH or SHOT point descriptor algorithms (section 2.2.5), with the centre points given by the previously detected keypoints.

**Constrained Sparse Matching**

Instead of matching the two keypoints with the smallest feature distances like in regular sparse matching, the keypoint pair with smallest feature distance that also adheres to a plane constraint is chosen. Where keypoints on point cloud $\mathcal{X}$ are matched to those on point cloud $\mathcal{Y}$, the plane constraint makes sure that the
keypoint positions, \( p_X \) and \( p_Y \), and the rotation \( R_{\gamma \leftarrow \chi} \) that a keypoint match gives rise to, are consistent with the correspondence between the two planes \( \pi_\chi \) and \( \pi_\gamma \). The rotation \( R_{\gamma \leftarrow \chi} \) is calculated by

\[
R_{\gamma \leftarrow \chi} = R_{\gamma \leftarrow \pi_Y} R_{\chi \leftarrow \pi_X}^{-1}
\]

where \( R_{\gamma \leftarrow \pi_Y} \) and \( R_{\chi \leftarrow \pi_X} \) describe the local reference frame of the keypoints. If calculating a repeatable local reference frame is not part of the feature description algorithm the method described in [Petrelli and Di Stefano, 2011] can be used. A keypoint pair is considered consistent with the plane constraint if:

\[
0 < \tau_R < \left( R_{\gamma \leftarrow \chi} n_{\pi_\chi} \right)^T n_{\pi_\gamma} \quad \text{and} \quad \tau_p > \left| n_{\pi_\chi}^T p_\chi + d_{\pi_\chi} - n_{\pi_\gamma}^T p_\gamma + d_{\pi_\gamma} \right|,
\]

where \( \tau_R \) and \( \tau_p \) are threshold parameters, \( n_{\pi_\chi} \) and \( n_{\pi_\gamma} \) are the normals of \( \pi_\chi \) and \( \pi_\gamma \), and \( d_{\pi_\chi} \) and \( d_{\pi_\gamma} \) are the offset parameters such that the planes are given by

\[
\pi_\chi = \left\{ p \in \mathbb{R}^3 : n_{\pi_\chi}^T p + d_{\pi_\chi} = 0 \right\}
\]

\[
\pi_\gamma = \left\{ p \in \mathbb{R}^3 : n_{\pi_\gamma}^T p + d_{\pi_\gamma} = 0 \right\}.
\]

**Correspondence Grouping**

**GCG** Correspondence grouping is done using the GCG algorithm, which is described in section 2.2.5.

**3D Hough Voting for Recognition** Correspondence grouping is done using the 3D hough voting for recognition algorithm, which is described in section 2.2.5.

**RANSAC with Correspondence Filtering** This method creates a single correspondence cluster with inliers as determined by RANSAC (section 2.2.5). Before RANSAC is performed, bad correspondences are removed by comparing virtual polygons created from small samples of correspondences, described in [Buch et al., 2013].

**Hypothesis Verification**

The reasons for correspondence grouping when doing object detection and registration of large point clouds are slightly different. When doing object detection a main motivation is that there could potentially be several instances of the same object in the scene, while in the registration case the grouping stage is mainly motivated by the risk of matching similar but different sections of the same scene in the two point clouds. Further, the hypothesis verification schemes presented by Aldoma et al. [2012] use viewpoint information when reasoning about potential object detections since the objects are meant to be detected in point clouds from range images. This means that these hypothesis verification methods are not applicable in the registration case. As a result of this thesis work, we therefore present two simple alternative methods.
3.1 Implementation

Naive Correspondence Counting The hypothesis corresponding to the cluster that contains the largest amount of correspondences is chosen. This method can not take hypotheses from plane matching into account.

Point-Error Scores Each hypothesis transformation $T_h$ is given a score according to the sum of errors between nearest match point-pairs after the first point cloud has been transformed by $T_h$. The Huber error metric is used to score the distance between two points. After scoring the transformations, the highest scoring hypothesis is chosen. This is very similar to the verification step of the SAC-IA algorithm presented in Rusu et al. [2009], where the same scoring mechanism is used together with a threshold to define the stopping criterion of a sample consensus algorithm.

Fine Alignment

The two point clouds are first sub-sampled using the normal-space sampling method described in [Rusinkiewicz and Levoy, 2001]. Then, ICP with a point-to-plane error metric is used to finely align the point clouds.

3.1.2 Pose Re-Optimization

Due to the wishes of Volumental, pose re-optimization is studied in the context of a single-session dense RGB-D SLAM algorithm with frame-to-TSDF tracking based on the work in [Bylow et al., 2013]. Since frames are tracked directly against the TSDF, and not against a virtual frame as in Newcombe et al. [2011], the information matrices returned by the frame registration algorithm refer to the frame-to-map transformation instead of the frame-to-previous-frame transformation. This has implications for how to set up the pose-graph when doing pose-graph optimization. Below we present four methods for doing pose re-optimization; re-tracking from estimates, two-map loop-closure, two-map loop-closure with cross-map building, and two-map loop-closure with edge-pruning, which are results from the work done on this thesis.

Re-Tracking From Estimates

The simplest form of pose re-optimization, is done by re-tracking each camera pose starting at the aligned set of poses from the registration stage. The two original sets of poses are first registered to each other and then regular frame-to-TSDF tracking is done, starting from the registered poses, sequentially building and tracking against a TSDF with the frames from both sessions.

Two-Map Loop-Closure

For sessions $A$ and $B$, we have the two sets of poses

$T^A_{1:N} = [T_{A←1}, T_{A←2}, \ldots, T_{A←N}]$, and

$T^B_{N+1:M} = [T_{B←N+1}, T_{B←N+2}, \ldots, T_{B←M}]$.

From the registration step we also have $T_{G←A} = T_{B←A}$ and $T_{G←B} = I$, which are added to the graph as virtual nodes representing the two maps.
We sequentially track the frames belonging to session $A$ as in re-tracking from estimates, giving the pose observations $z_{A,1}, \ldots, z_{A,N}$ with information matrices $\Omega_{A,1}, \ldots, \Omega_{A,N}$, relating each frame to map $A$. The corresponding edges $(z_{A,1}, \Omega_{A,1}), \ldots, (z_{A,N}, \Omega_{A,N})$ and poses $T^A_{1:N}$ are then added to the pose-graph. We then track the frames belonging to session $B$ against the previously built TSDF from $A$, without fusing in any new data into the TSDF. This gives us the observation edges $(z_{A,1}, \Omega_{A,N+1}), \ldots, (z_{A,N}, \Omega_{A,M})$, which are added to the pose-graph. The poses from session $B$ are initialized by aligning the poses $T^B_{N+1:M}$ to map $A$ with $T_{A\leftarrow B}^1 = T_{B\leftarrow A}^1$ from the registration stage.

The whole process is then repeated, where the frames belonging to session $B$ are instead used to build the TSDF, resulting in a pose-graph in the form given in figure 3.3.

After pose-graph optimization, we have the two sets of optimized poses $T^A_{1:N}$ and $T^B_{N+1:M}$. The final optimized set of poses $T^G_{1:M} = [T^G_{A\leftarrow 1}, \ldots, T^G_{B\leftarrow M}]$ is then found by setting

$$T^G_{A\leftarrow i} = \begin{cases} T^A_{A\leftarrow i} & \text{if } 0 < i \leq N, \\ T^B_{B\leftarrow i} & \text{if } N < i \leq M. \end{cases}$$

**Two-Map Loop-Closure With Edge-Pruning**

In regular two-map loop-closure, no special handling is done for edges where the frame-to-other-session-map optimization does not converge. In this scheme we only add edges for frame tracking optimizations that converge. Convergence is decided by running extra optimization steps and comparing the distance between the poses to a threshold. We select a threshold of $0.5 \text{mm}$ between the translation components of the poses.

**Two-Map Loop-Closure With Cross-Map Building**

The only difference between doing two-map loop-closure with cross-map building and regular two-map loop-closure, is that the TSDF for map $A$ (and $B$ respectively) is built with the frames from both sessions but starting with the frames from session $A$ (or $B$ respectively). This method extends the two-map loop-closure with edge-pruning method with cross-map building.
3.1 Implementation

Figure 3.3: Two-Map Pose-Graph
The nodes in the graph consist of the camera poses (blue) $T_1, T_2, \ldots, T_M$, and the virtual nodes (gray) $T_A$ and $T_B$ representing the two maps. All edges in the graph are between camera poses and map poses, and represent the relative pose constraints from frame-to-map tracking. An edge between a camera pose and a map that belong to the same session is marked by a solid line, and an edge from different sessions by a dashed line.
3.1.3 Fusion

Once all the poses are registered to each other (and potentially re-optimized), the data from all the frames is fused together. This is done by running a TSDF based reconstruction with the composed pose-set as input. In the case where no pose re-optimization is run, this is equivalent to doing a weighted average, using the per-voxel weights from sequential TSDF building, of the aligned TSDF’s from the two sessions and then meshing the result.

3.1.4 Repositioned Objects In-Between Scans

To test the system’s ability to handle fusing objects together that have been moved in between scans, we consider the simple scenario where the only things visible in both recordings are the object to be scanned and the planar surface that it is placed on. With the exception of some special handling of the planar surface, the same system can be used to reconstruct a repositioned object as is used for a static scene. The handling of the planar surface in the different steps of the system is explained below.

Registration

Since the object pose relative the ground plane is not consistent between the two recordings, the early-integration pipeline is not usable for this problem. Instead, the late-integration pipeline is used with the modification that the ground plane is removed before all other steps and that no plane matching is done. The parameters of the detected ground planes are also outputted to be used in pose re-optimization and fusion. A depiction of the planeless registration pipeline can be seen in figure 3.4.

Pose Re-Optimization and Fusion

In the pose re-optimization and fusion steps, the ground planes are removed from each frame before any further processing is done, such that the rest of the system can act as if a freely floating object had been scanned in two sessions.
3.1 Implementation

Figure 3.4: Planeless Registration Pipeline

The figure above depicts an overview of the point cloud registration pipeline for registering repositioned objects, with the arrows showing the flow of the data between different pipeline steps. Hypotheses transformations are generated by sparse keypoint based matching between the point clouds after the ground planes have been removed from both scenes. The best hypothesis is then chosen and passed on to the fine alignment step. Both the aligning transformation and ground plane parameters are output to later steps.
3.2 Evaluation

We evaluate the above described system using two datasets, the TUM RGB-D dataset and the partial scan dataset, which are described below. The registration part of the system is evaluated on the TUM RGB-D dataset and the whole system, including pose re-optimization and fusion, is evaluated together on the partial scan dataset.

3.2.1 Metrics

We use the metrics for evaluating object pose estimation given in [Petricek, 2012], as their notion of object pose corresponds to the sought transformation $T_{B,A}$ in our case. The error metrics used are pose error $d: \mathbb{SE}(3) \times \mathbb{SE}(3) \mapsto \mathbb{R}$, rotation error $r_{error}: \mathbb{SO}(3) \times \mathbb{SO}(3) \mapsto [0, 180\degree]$, and translation error $t_{error}: \mathbb{R}^3 \times \mathbb{R}^3 \mapsto \mathbb{R}$, given by:

$$
t_{error}(t_1, t_2) = \|t_1 - t_2\|_2 \\
r_{error}(R_1, R_2) = \alpha(R_1^T R_2) \\
d(T_1, T_2) = \max\left(\frac{r_{error}(R_1, R_2)}{z_R}, \frac{t_{error}(t_1, t_2)}{z_t}\right),
$$

where $\alpha(R)$ is the angle component of the angle-axis representation of $R$. $z_R$ and $z_t$ are scaling factors that determine the relative cost of errors in translation to errors in rotation. We choose to set $z_R = 1$ for simplicity. $z_t$ is determined per dataset in the fine alignment experiment explained in section 3.2.4. The translation, rotation, or pose error of an alignment refers to the error between the found alignment transformation and the ground truth.

3.2.2 TUM RGB-D Dataset

Eight datasets are chosen from the repository of RGB-D recordings available at [Sturm et al., 2012]. We choose the datasets, listed in table 3.1, that most resemble the object reconstruction scenario. All of the recordings have corresponding ground truth pose trajectories gathered by a motion capture system. The recordings were collected primarily to be used for performance evaluation of SLAM systems and some are therefore shaky and contain many camera poses that are directed away from the object to be scanned.

Each recording is divided into two equally long parts and the two sets of poses are put through a sub-routine, provided by Volumental, that re-aligns the poses such that the z-axis is made roughly parallel with the real world up-axis. The sub-routine is applied to the two sets of poses separately, resulting in that they are put into different reference frames. KinectFusion [Newcombe et al., 2011], available in the PCL library, is used to reconstruct the scenes from the recordings. It is slightly modified to take the ground truth poses as input and reconstruct the scenes directly without tracking. After reconstruction, we calculate vertex normals as the area weighted average of the face normals of the surrounding triangles.
The vertices with normals are kept and the result is two point clouds with normals, that we label from and to, for each recording. Renderings of the resulting models can be seen in appendix A.1.

In particular, it is worth noting that the reconstructions for the coke and both the flower bouquet recordings (figures A.4 to A.6) look particularly inaccurate, and that the reconstructions from the cabinet (figure A.7) recording shows a very large degree of symmetry.

An example of the renderings of the TUM RGB-D dataset can be seen in figure 3.5, where we see renderings of the Freiburg1: teddy reconstructions.

<table>
<thead>
<tr>
<th>Recording Name</th>
<th>Number of frames</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freiburg1: teddy</td>
<td>1400</td>
</tr>
<tr>
<td>Freiburg1: xyz</td>
<td>790</td>
</tr>
<tr>
<td>Freiburg1: plant</td>
<td>1120</td>
</tr>
<tr>
<td>Freiburg2: coke</td>
<td>2472</td>
</tr>
<tr>
<td>Freiburg2: flowerbouquet</td>
<td>2859</td>
</tr>
<tr>
<td>Freiburg2: flowerbouquet-brownbackground</td>
<td>2234</td>
</tr>
<tr>
<td>Freiburg3: cabinet</td>
<td>1108</td>
</tr>
<tr>
<td>Freiburg3: teddy</td>
<td>2321</td>
</tr>
</tbody>
</table>

**Table 3.1: TUM RGB-D Recordings**

The data from the TUM RGB-D recordings, used for evaluation of the registration pipeline.

**Figure 3.5: Freiburg1: teddy Renderings.**

Renderings of the From (left) and To (right) reconstructions from the Freiburg1: teddy recording.
Ground Truth

The registration pipeline outputs a transformation $T_{B←A}$ that is meant to be applied to point cloud $\mathcal{X}$, in order to bring it into alignment with point cloud $\mathcal{Y}$. The sets of poses $T_{1:N}^A$ and $T_{1:M}^B$ together make up the ground truth poses from the motion capture system. As described above, these two sets are put through a subroutine, which transforms them by two different rigid transformations, resulting in the pose sets $\tilde{T}_{1:N}^A$ and $\tilde{T}_{1:M}^B$, which are used in the reconstructions producing $\mathcal{X}$ and $\mathcal{Y}$. The ground truth transformation $T_{B←A}^{gt}$ is then found by:

$$T_{B←A}^{gt} = \tilde{T}_{B}^{-1}(T_{B}^{-1})^{-1}$$

### 3.2.3 Partial Reconstruction Dataset

We record two separate recordings per scene, for ten different scenes. Two small objects, a statue and a foot model, were placed in different surroundings, and two larger objects, a bag and a chair, were scanned in one place. The scenes were chosen to represent realistic scenarios for when a user might scan an object in two sessions, either to amend an incomplete scan or to scan an object from all angles. The small objects were scanned on a clean table, a cluttered table, and a cluttered table with several identical objects to enable analysis of the system’s performance on a variety of scenarios. An object on a clean table means that there are fewer geometrically interesting shapes in the scene that can be matched between the two scans in the sparse matching stage, as well as less well constrained when doing optimization problems in fine registration and pose re-optimization. Duplicate objects in the scene test the registration pipeline’s ability to handle strong false sparse matches.

The recordings of the small objects were reconstructed with a single-session system based on [Newcombe et al., 2011] and the chair and bag scenes were reconstructed with a system based on [Endres et al., 2012]. The point clouds with normals are constructed from the mesh models in the same way as for the TUM RGB-D dataset and the two recordings of each scene are here also given the labels from and to.

A summary of the recordings in the dataset can be found in table 3.2 and renderings of the models in A.2.

In particular, it is worth noting that the KinectFusion [Newcombe et al., 2011] single-session reconstruction failed and produced a warped foot for the to recording of the foot: clean (figure A.9) scene. It is also worth noting the varying level of overlap between the pairs of reconstructed models. For instance, both of the cluttered (figures A.11 and A.12) scenes have much more overlap between the two recordings than the Chair scene does. We can also observe that in for instance the larger foot reconstructions cluttered and duplicates, reconstruction errors occur at the top of the foot model, appearing as an elongation of the top edge away from where the RGB-D video recording was captured.
An example of the renderings of the *partial reconstruction dataset* can be seen in figure 3.6, where we see renderings of the *foot:cluttered* reconstructions.

<table>
<thead>
<tr>
<th>Scene Name</th>
<th>Number of frames</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foot: Clean</td>
<td>499 + 810</td>
</tr>
<tr>
<td>Bust: Clean</td>
<td>431 + 438</td>
</tr>
<tr>
<td>Foot: Cluttered</td>
<td>523 + 525</td>
</tr>
<tr>
<td>Bust: Cluttered</td>
<td>821 + 465</td>
</tr>
<tr>
<td>Foot: Duplicates</td>
<td>451 + 582</td>
</tr>
<tr>
<td>Bust: Duplicates</td>
<td>724 + 511</td>
</tr>
<tr>
<td>Foot: Repositioned</td>
<td>748 + 671</td>
</tr>
<tr>
<td>Bust: Repositioned</td>
<td>957 + 569</td>
</tr>
<tr>
<td>Chair</td>
<td>570 + 432</td>
</tr>
<tr>
<td>Bag</td>
<td>600 + 698</td>
</tr>
</tbody>
</table>

*Table 3.2: Partial Reconstructions Dataset*

The scenes in the Partial Reconstructions Dataset used for evaluation of the full system. The number of frames is given in the format \( n_{\text{from}} + n_{\text{to}} \), where \( n_{\text{from}} \) and \( n_{\text{to}} \) are the number of frames in the From and To recordings respectively.

![Figure 3.6: Foot:Cluttered Renderings.](image)

Renderings of the From (left) and To (right) reconstructions of the Foot:Cluttered scene.

### 3.2.4 Experiments

The goal is to answer the question: *what is the best way to do multi-session dense reconstruction?* We would therefore like to know, for the different steps, which methods are best, why, and under which circumstances. Most of the implementation focus is on the rough alignment part of the system, which is therefore also the main focus of the evaluation.
We first look broadly at what combination of methods can give good performance for rough alignment, and then for a smaller subset we look at how good performance we can attain under two different objectives for parameter tuning. We also analyze the chosen method for fine alignment to help find per-dataset bounds on the rough alignment errors to determine success or failure of that step, as well as the rotation to translation ratio necessary to compute the combined pose error. Finally, we analyze the pose re-optimization and fusion together by doing full two-session reconstruction on the partial reconstructions dataset using the best performing registration method.

**Rough Alignment Broad Search**

The rough alignment part of the pipeline is run on the eight recordings in the TUM RGB-D dataset, with 44 different component combinations to the late-integration pipeline (the early-integration pipeline was not conceived at the time of this experiment). There are 48 different possible combinations to this pipeline, but 4 may be discarded as pipelines without plane matching that use RANSAC correspondence grouping only produce one hypothesis, and therefore don’t need hypothesis testing. To make sure any particular configuration is not discarded due to poorly set parameters, the most important parameters and those for which the effects are uncertain are varied over a range, resulting in that over 1300 different configurations of the system are run on each recording. The median rotation error, translation error and computation time over the eight datasets are observed for each configuration. The aim of the broad search experiment is to restrict the rest of the analysis to pipeline variations that are deemed both fast enough and potentially performant.

**Rough Alignment Performance Analysis**

To narrow down the search space, we select the pipeline variations that have a median translation error below 15cm, rotation error below 15°, and computation time below 5 seconds on some parameter combination in the broad search experiment. We then do a parameter tuning for the remaining methods by varying the parameters in a range around the best performing configurations from the previous experiment and observe the median pose error and number of successful rough alignments for each configuration. For each of the methods, the whole registration pipeline is then run on each recording separately using the best parameter configurations under both pose error and number of successes.

**Fine Alignment**

We would like to test the convergence properties of the flavor of ICP that is described in 3.1.1, in order to assist in the analysis of rough alignment performance. We first align the two point clouds to each other using the ground truth transformation. We then create a series of perturbation transformations by sampling the six parameters, three Euler-angles and three translation components, of a rigid transformation over a regular grid. For each combination of angles and translation components, the first point cloud is transformed with the perturbation transformation, simulating a roughly aligned point cloud. The rotation and
translation errors are observed for both the input and final transformation.

We would like to find the scale parameter, $z_t$, needed to calculate pose error, and a threshold, $\tau$ on pose error for rough alignment, $e_{\text{rough}}$, to be considered successful. The purpose of using the pose error metric is to be able to predict how good the fine alignment will be from the pose error of the rough alignment. We therefore search for the scale parameter $z_t$ that makes $e_{\text{rough}}$ be a good predictor of the pose error after fine alignment $e_{\text{fine}}$. Further, we assume that the ICP algorithm has some basin of convergence for each dataset such that, given a good choice of $z_t$, there is some $e_{\text{rough}}$ above which fine alignment starts to fail. This gives rise to the following simple scheme for finding $z_t$ and $\tau$ for a dataset:

1. For each perturbation transformation, observe the pair $(e_{\text{rough}}, e_{\text{fine}})$.
2. Choose $z_t$ such that $e_{\text{rough}}$ is the best possible linear predictor of $e_{\text{fine}}$.
3. Sort the observation pairs in ascending order in $e_{\text{rough}}$.
4. Starting from zero, taking small steps in $e_{\text{rough}}$, observe the largest $e_{\text{fine}}$ within the step. If this is more than $\alpha = 2$ times larger than the largest previously seen $e_{\text{fine}}$, choose current position as threshold $\tau$.

**Pose Re-Optimization and Fusion**

We evaluate the whole system, but in particular pose re-optimization and fusion, on the partial reconstructions dataset, by first aligning the two recordings to each other using the best performing registration pipeline, and then either fusing them directly or first re-optimizing the poses using the methods described in 3.1.2. We then perform a qualitative evaluation of the resulting fused models to compare the performance of the different re-optimization methods and help in drawing conclusions concerning the effectiveness of the whole system.

Each fused model is compared to its corresponding partial models according to whether or not the fused model has:

**Successful Alignment** The data from the two partial scans has been aligned in a roughly correct manner such that the result is clearly a model of the same scene as the partial reconstructions.

**Reduced Holes** The fused model has removed holes, or is extended, in a way that could not be achieved by simply choosing one of the two partial reconstructions.

**Added Holes** The fused model has holes that are not present in any of the two partial reconstructions.

**Reduced Defects** The fused model has reduced one or more pronounced reconstruction defects that are present in at least one of the two partial reconstructions. Reconstruction defects are things such as extended edges and general shape deformations such as bending.
**Added Defects** The fused model contains one or more enhanced or new reconstruction defect that is not present or considerably less pronounced in both of the two partial reconstructions.

The fused models from different methods are compared to each other by classifying each fused model with one of the relative labels *good*, *mediocre*, or *bad*. All labels are not necessary assigned to models of a given scene. The labels regard visual aspects of the result and are awarded on a relative basis compared to the other fused models. The default label *mediocre* is given in all cases where an effect can not be deemed pronounced enough in relation to the other fused models.

The relative labels are awarded according to how the following problems are handled:

**Fusion Edges** Edges may occur due to imperfect alignment of the data from the two scans.

**Main Object Deformations** The main object in the scene, i.e. the foot, bust, chair, or bag, may have differing degrees of deformation compared to the real-life model.
In this chapter we present the results from the evaluation experiments described in section 3.2.4. We begin by presenting the results from the rough alignment broad search and rough alignment performance analysis experiments where the performance of the different rough alignment pipelines is evaluated. We then present results of the fine alignment experiment, where the convergence properties of the used fine alignment method on the TUM RGB-D dataset is analysed. We end by presenting a qualitative analysis of the results of the whole system, with a focus on the pose re-optimization and fusion steps.

For brevity, the registration pipelines are abbreviated with the syntax: "Filter"-"Keypoint Detection"-"Sparse Matching"-"Correspondence Grouping"-"Hypothesis Verification", with shortened names of each step variation in place of the step names. For example, Plane-ISS-SHOT-GCG-PES is the pipeline with plane filtering and matching, ISS point detectors, sparse matching with SHOT point descriptors, geometric consistency grouping, and hypothesis verification via point-error scores.

### 4.1 Rough Alignment Broad Search

In the broad search section of the rough alignment analysis we gather data to help reduce the number of variations on the registration pipeline, to allow for a more thorough analysis of the most promising variations in the performance analysis section.

The plots in figures 4.1 to 4.3 show the median results over the TUM RGB-D dataset for execution time, translation error and rotation error. Each result gives horizontal lines in the columns naming the pipeline step variations that are used
in that run. Each color represents a pipeline step, and columns of the same color represent variations of the same pipeline step, such that the sum of lines in a single color equals the number of runs in the experiment.

Most step variations are present on all levels of performance. It is evident however, that the runs using ISS and SHOT execute faster than those using *uniform sampling* and FPFH. It is also clear that the usage of *point-error scores* (PES) gives better median accuracy than using *naive correspondence counting* (NCC). These points are also made clear in table 4.1, which lists the results from the pipelines variations with execution speed below 5s, translation error below 15cm, and rotation error below 15° in median. With these constraints, all remaining pipeline variations use ISS keypoint detection and SHOT based sparse matching. In all but one trivial case (RANSAC only generates one sparse hypothesis and no planes are matched with a pass through filter), the remaining configurations use the PES hypothesis verification scheme.

All the runs in this experiment are different variations of the *late-integration pipeline* since the *early-integration pipeline* was not conceived at the time of this experiment.

*Figure 4.1: Timing results for all runs plotted as standing rug plots.* Each pipeline step has its own color, and results shown are the median over all recordings. A single horizontal line represents the result for one pipeline configuration, such that each column represents a rug plot of the execution times of all configurations using a particular step variation.
Figure 4.2: Translation error results for all runs plotted as standing rug plots. Each pipeline step has its own color, and results shown are the median over all recordings. A single horizontal line represents the result for one pipeline configuration, such that each column represents a rug plot of the translation errors of all configurations using a particular step variation.

Table 4.1: Rough Alignment Broad Search: Top Performance.
The best median results over the TUM RGB-D Datasets, filtered for configurations with execution speed below 5s, translation error below 15cm, and rotation error below 15°.
Figure 4.3: Rotation error results for all runs plotted as standing rug plots. Each pipeline step has its own color, and results shown are the median over all recordings. A single horizontal line represents the result for one pipeline configuration, such that each column represents a rug plot of the rotation errors of all configurations using a particular step variation.
4.2 Rough Alignment Performance Analysis

In the performance analysis section of the rough alignment analysis, we do a more thorough search of the parameter space of the best performing pipelines from section 4.1 to optimize for alignment performance. In addition to the best pipelines from section 4.1, we also analyze the Plane-ISS-SHOTC-GCG-PES configuration on the early-integration pipeline. SHOTC is constrained sparse matching using the SHOT point descriptor.

We first present the aggregate results for each pipeline in the form of histograms of the number of successful alignments and median pose errors on the TUM RGB-D Dataset for each of the pipelines. We then present the performance of the best performing (in terms of median pose error) parameter configurations for each pipeline variation on each of the recordings separately.

4.2.1 Aggregate results

In figures 4.4 to 4.9, we present the histograms of the number of successful alignments and normalized cumulative histograms of median pose errors over the TUM RGB-D dataset. The number of successes over the dataset can be seen as a proxy for overall robustness, and the median pose error as a robust measure of general accuracy. A rough alignment is considered a success if the pose error after alignment is below the scene dependent threshold, \( \tau \), which is determined in the fine alignment experiment. The best pipeline would then have more runs than the others with many successes, and would show high accuracy for a comparatively large proportion of the total runs.

It is clear, when comparing the histograms of all the examined pipelines, that the Plane-ISS-SHOTC-GCG-PES pipeline has the best distribution of number of successful alignments for the ensemble of runs performed in this experiment. It has more runs with four or five successes and less runs with only zero or one success than any of the other pipelines.

In terms of the distributions of median pose error, which is defined in section 3.2.1, the Plane-ISS-SHOTC-GCG-PES (figure 4.9) and Plane-ISS-SHOT-Hough-PES (figure 4.7) pipelines clearly perform the best. The Plane-ISS-SHOTC-GCG-PES pipeline has the most runs with less than 8 median pose error, but the Plane-ISS-SHOT-Hough-PES pipeline clearly has the best distribution of errors for the remaining range, with 80% of runs with median pose errors of less than 20.

The two pipelines using RANSAC clearly show the worst performance in terms of number of successful alignments, with very few runs with three or more successes. In particular, the simplest pipeline Pass-ISS-SHOT-RANSAC-NCC has no runs with four or more successes.

In table 4.2 we present the number of successes and the median pose error for the best performing (in terms of median pose error) parameter configuration for each of the pipelines. We can first note that all but the Plane-ISS-SHOT-RANSAC-PES pipeline attain their maximum number of successes with the same parameter
configurations that give the lowest median pose error. We may also observe that the Plane-ISS-SHOTC-GCG-PES pipeline reaches the lowest median pose error of all the pipelines.

Another interesting observation is that the Plane-ISS-SHOT-GCG-PES and the Pass-ISS-SHOT-GCG-PES pipelines both have the exact same minimum median pose error, indicating that the plane hypothesis is not used for the most accurate alignments. This identical minimum was reached with identical values of all shared parameters.

(a) Histogram of the number of successful rough alignments for each run of the pipeline over the TUM RGB-D Dataset.

(b) Normalized cumulative histogram of the median pose errors after rough alignments for each run of the pipeline over the TUM RGB-D Dataset.

**Figure 4.4:** Plane-ISS-SHOT-GCG-PES performance. Histograms summing up the performance of the Plane-ISS-SHOT-GCG-PES pipelines. Each run represents one parameter configuration.
4.2 Rough Alignment Performance Analysis

(a) Histogram of the number of successful rough alignments for each run of the pipeline over the TUM RGB-D Dataset.

(b) Normalized cumulative histogram of the median pose errors after rough alignments for each run of the pipeline over the TUM RGB-D Dataset.

Figure 4.5: Pass-ISS-SHOT-GCG-PES performance. Histograms summing up the performance of the Pass-ISS-SHOT-GCG-PES pipelines. Each run represents one parameter configuration.

(a) Histogram of the number of successful rough alignments for each run of the pipeline over the TUM RGB-D Dataset.

(b) Normalized cumulative histogram of the median pose errors after rough alignments for each run of the pipeline over the TUM RGB-D Dataset.

Figure 4.6: Pass-ISS-SHOT-RANSAC-NCC performance. Histograms summing up the performance of the Pass-ISS-SHOT-RANSAC-NCC pipelines. Each run represents one parameter configuration.
44 Results

(a) Histogram of the number of successful rough alignments for each run of the pipeline over the TUM RGB-D Dataset.

(b) Normalized cumulative histogram of the median pose errors after rough alignments for each run of the pipeline over the TUM RGB-D Dataset.

Figure 4.7: Plane-ISS-SHOT-Hough-PES performance. Histograms summing up the performance of the Plane-ISS-SHOT-Hough-PES pipelines. Each run represents on parameter configuration.

(a) Histogram of the number of successful rough alignments for each run of the pipeline over the TUM RGB-D Dataset.

(b) Normalized cumulative histogram of the median pose errors after rough alignments for each run of the pipeline over the TUM RGB-D Dataset.

Figure 4.8: Plane-ISS-SHOT-RANSAC-PES performance. Histograms summing up the performance of the Plane-ISS-SHOT-RANSAC-PES pipelines. Each run represents on parameter configuration.
4.2 Rough Alignment Performance Analysis

(a) Histogram of the number of successful rough alignments for each run of the pipeline over the TUM RGB-D Dataset.

(b) Normalized cumulative histogram of the median pose errors after rough alignments for each run of the pipeline over the TUM RGB-D Dataset.

Figure 4.9: Plane-ISS-SHOTC-GCG-PES performance. Histograms summing up the performance of the Plane-ISS-SHOTC-GCG-PES pipelines. Each run represents one parameter configuration.
<table>
<thead>
<tr>
<th>Pipeline</th>
<th>Number of Successes</th>
<th>Median Pose Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plane-ISS-SHOT-GCG-PES</td>
<td>5</td>
<td>3,750</td>
</tr>
<tr>
<td>Pass-ISS-SHOT-GCG-PES</td>
<td>5</td>
<td>3,750</td>
</tr>
<tr>
<td>Pass-ISS-SHOT-RANSAC-NCC</td>
<td>3</td>
<td>8,065</td>
</tr>
<tr>
<td>Plane-ISS-SHOT-Hough-PES</td>
<td>5</td>
<td>4,919</td>
</tr>
<tr>
<td>Plane-ISS-SHOT-RANSAC-PES</td>
<td>1</td>
<td>8,462</td>
</tr>
<tr>
<td>Plane-ISS-SHOTC-GCG-PES</td>
<td>5</td>
<td>2,216</td>
</tr>
</tbody>
</table>

Table 4.2: Lowest Median Pose Error.
The number of successful alignments and median pose error for the parameter configuration with the lowest median pose error for each pipeline.

4.2.2 Best Results Per Dataset

In tables 4.3 to 4.9 we present the per-dataset results of the best performing pipeline configurations.

All pipelines completely failed to correctly align the Freiburg2: coke and Freiburg3: cabinet datasets, with pose errors close to or above 180. On a large scale, these datasets have a high degree of shape symmetry, and on a smaller scale lack repeatable geometric features.

Excluding these failed datasets, the picture presented in 4.2.1 is strengthened. The Plane-ISS-SHOT-GCG-PES pipeline has the best top performance in general, with the most consistently low pose errors.

As was noted in section 4.2.1, the Plane-ISS-SHOT-GCG-PES and the Pass-ISS-SHOT-GCG-PES pipelines both have the same results for the best performing parameter choices. Again, this indicates that the plane is only used when the sparse pipeline performs well.

<table>
<thead>
<tr>
<th>Pipeline</th>
<th>Pose Error</th>
<th>Trans. Error [m]</th>
<th>Rot. Error [°]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plane-ISS-SHOT-GCG-PES</td>
<td>3.782</td>
<td>0.045</td>
<td>3.781</td>
</tr>
<tr>
<td>Pass-ISS-SHOT-GCG-PES</td>
<td>3.782</td>
<td>0.045</td>
<td>3.781</td>
</tr>
<tr>
<td>Pass-ISS-SHOT-RANSAC-NCC</td>
<td>29.721</td>
<td>0.636</td>
<td>29.721</td>
</tr>
<tr>
<td>Plane-ISS-SHOT-Hough-PES</td>
<td>16.639</td>
<td>0.287</td>
<td>16.639</td>
</tr>
<tr>
<td>Plane-ISS-SHOT-RANSAC-PES</td>
<td>29.201</td>
<td>0.752</td>
<td>29.202</td>
</tr>
<tr>
<td>Plane-ISS-SHOTC-GCG-PES</td>
<td><strong>1.929</strong></td>
<td><strong>0.035</strong></td>
<td><strong>1.929</strong></td>
</tr>
</tbody>
</table>

Table 4.3: Freiburg1: teddy Performance.
The results when using the parameters with the lowest median pose error for each of the analyzed pipelines on the Freiburg1: teddy recording.
## 4.2 Rough Alignment Performance Analysis

### Table 4.4: Freiburg1: xyz Performance.
The results when using the parameters with the lowest median pose error for each of the analyzed pipelines on the Freiburg1: xyz recording.

<table>
<thead>
<tr>
<th>Pipeline</th>
<th>Pose Error</th>
<th>Trans. Error [m]</th>
<th>Rot. Error [°]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plane-ISS-SHOT-GCG-PES</td>
<td>3.719</td>
<td>0.044</td>
<td>3.719</td>
</tr>
<tr>
<td>Pass-ISS-SHOT-GCG-PES</td>
<td>3.719</td>
<td>0.044</td>
<td>3.719</td>
</tr>
<tr>
<td>Pass-ISS-SHOT-RANSAC-NCC</td>
<td>9.515</td>
<td>0.190</td>
<td>6.761</td>
</tr>
<tr>
<td>Plane-ISS-SHOT-Hough-PES</td>
<td><strong>0.919</strong></td>
<td><strong>0.018</strong></td>
<td><strong>0.667</strong></td>
</tr>
<tr>
<td>Plane-ISS-SHOT-RANSAC-PES</td>
<td>3.809</td>
<td>0.076</td>
<td>2.676</td>
</tr>
<tr>
<td>Plane-ISS-SHOTC-GCG-PES</td>
<td>1.735</td>
<td>0.035</td>
<td>1.411</td>
</tr>
</tbody>
</table>

### Table 4.5: Freiburg1: plant Performance.
The results when using the parameters with the lowest median pose error for each of the analyzed pipelines on the Freiburg1: plant recording.

<table>
<thead>
<tr>
<th>Pipeline</th>
<th>Pose Error</th>
<th>Trans. Error [m]</th>
<th>Rot. Error [°]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plane-ISS-SHOT-GCG-PES</td>
<td><strong>3.276</strong></td>
<td><strong>0.068</strong></td>
<td>1.921</td>
</tr>
<tr>
<td>Pass-ISS-SHOT-GCG-PES</td>
<td><strong>3.276</strong></td>
<td><strong>0.068</strong></td>
<td>1.921</td>
</tr>
<tr>
<td>Pass-ISS-SHOT-RANSAC-NCC</td>
<td>6.060</td>
<td>0.126</td>
<td>5.977</td>
</tr>
<tr>
<td>Plane-ISS-SHOT-Hough-PES</td>
<td>8.311</td>
<td>0.085</td>
<td>8.311</td>
</tr>
<tr>
<td>Plane-ISS-SHOT-RANSAC-PES</td>
<td>17.766</td>
<td>0.253</td>
<td>17.767</td>
</tr>
<tr>
<td>Plane-ISS-SHOTC-GCG-PES</td>
<td>14.248</td>
<td>0.137</td>
<td>14.248</td>
</tr>
</tbody>
</table>

### Table 4.6: Freiburg2: coke Performance.
The results when using the parameters with the lowest median pose error for each of the analyzed pipelines on the Freiburg2: coke recording. No results are highlighted as all pipelines in effect failed.

<table>
<thead>
<tr>
<th>Pipeline</th>
<th>Pose Error</th>
<th>Trans. Error [m]</th>
<th>Rot. Error [°]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plane-ISS-SHOT-GCG-PES</td>
<td>227.809</td>
<td>1.529</td>
<td>194.661</td>
</tr>
<tr>
<td>Pass-ISS-SHOT-GCG-PES</td>
<td>227.809</td>
<td>1.529</td>
<td>194.661</td>
</tr>
<tr>
<td>Pass-ISS-SHOT-RANSAC-NCC</td>
<td>243.999</td>
<td>1.638</td>
<td>178.912</td>
</tr>
<tr>
<td>Plane-ISS-SHOT-Hough-PES</td>
<td>211.738</td>
<td>1.421</td>
<td>174.645</td>
</tr>
<tr>
<td>Plane-ISS-SHOT-RANSAC-PES</td>
<td>238.457</td>
<td>1.600</td>
<td>180.627</td>
</tr>
<tr>
<td>Plane-ISS-SHOTC-GCG-PES</td>
<td>238.245</td>
<td>1.599</td>
<td>174.214</td>
</tr>
<tr>
<td>Pipeline</td>
<td>Pose Error</td>
<td>Trans. Error [m]</td>
<td>Rot. Error [°]</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>------------</td>
<td>-----------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Plane-ISS-SHOT-GCG-PES</td>
<td>16.730</td>
<td>0.159</td>
<td>13.502</td>
</tr>
<tr>
<td>Pass-ISS-SHOT-GCG-PES</td>
<td>16.730</td>
<td>0.159</td>
<td>13.502</td>
</tr>
<tr>
<td>Pass-ISS-SHOT-RANSAC-NCC</td>
<td>2.691</td>
<td>0.026</td>
<td>2.434</td>
</tr>
<tr>
<td>Plane-ISS-SHOT-Hough-PES</td>
<td>11.377</td>
<td>0.108</td>
<td>4.279</td>
</tr>
<tr>
<td>Plane-ISS-SHOT-RANSAC-PES</td>
<td>5.659</td>
<td>0.054</td>
<td>1.506</td>
</tr>
<tr>
<td>Plane-ISS-SHOTC-GCG-PES</td>
<td>2.315</td>
<td>0.014</td>
<td>2.315</td>
</tr>
</tbody>
</table>

Table 4.7: Freiburg2: flowerbouquet Performance.
The results when using the parameters with the lowest median pose error for each of the analyzed pipelines on the Freiburg2: flowerbouquet recording.

<table>
<thead>
<tr>
<th>Pipeline</th>
<th>Pose Error</th>
<th>Trans. Error [m]</th>
<th>Rot. Error [°]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plane-ISS-SHOT-GCG-PES</td>
<td>3.515</td>
<td>0.078</td>
<td>2.587</td>
</tr>
<tr>
<td>Pass-ISS-SHOT-GCG-PES</td>
<td>3.515</td>
<td>0.078</td>
<td>2.587</td>
</tr>
<tr>
<td>Pass-ISS-SHOT-RANSAC-NCC</td>
<td>8.649</td>
<td>0.094827</td>
<td>8.649</td>
</tr>
<tr>
<td>Plane-ISS-SHOT-RANSAC-PES</td>
<td>21.052</td>
<td>0.46782</td>
<td>11.611</td>
</tr>
<tr>
<td>Plane-ISS-SHOTC-GCG-PES</td>
<td>8.997</td>
<td>0.199937</td>
<td>6.302</td>
</tr>
</tbody>
</table>

Table 4.8: Freiburg2: flowerbouquet-brownbackground Performance.
The results when using the parameters with the lowest median pose error for each of the analyzed pipelines on the Freiburg2: flowerbouquet-brownbackground recording.

<table>
<thead>
<tr>
<th>Pipeline</th>
<th>Pose Error</th>
<th>Trans. Error [m]</th>
<th>Rot. Error [°]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plane-ISS-SHOT-GCG-PES</td>
<td>176.73344</td>
<td>2.32544</td>
<td>168.767</td>
</tr>
<tr>
<td>Pass-ISS-SHOT-GCG-PES</td>
<td>176.73344</td>
<td>2.32544</td>
<td>168.767</td>
</tr>
<tr>
<td>Pass-ISS-SHOT-RANSAC-NCC</td>
<td>201.126</td>
<td>2.27882</td>
<td>201.126</td>
</tr>
<tr>
<td>Plane-ISS-SHOT-Hough-PES</td>
<td>178.12196</td>
<td>2.34371</td>
<td>177.4</td>
</tr>
<tr>
<td>Plane-ISS-SHOT-RANSAC-PES</td>
<td>182.092</td>
<td>2.34431</td>
<td>182.092</td>
</tr>
<tr>
<td>Plane-ISS-SHOTC-GCG-PES</td>
<td>193.1046</td>
<td>2.54085</td>
<td>179.654</td>
</tr>
</tbody>
</table>

Table 4.9: Freiburg3: cabinet Performance.
The results when using the parameters with the lowest median pose error for each of the analyzed pipelines on the Freiburg3: cabinet recording. No results are highlighted as all pipelines in effect failed.
### Table 4.10: Freiburg3: teddy Performance.

The results when using the parameters with the lowest median pose error for each of the analyzed pipelines on the Freiburg3: teddy recording.

<table>
<thead>
<tr>
<th>Pipeline</th>
<th>Pose Error</th>
<th>Trans. Error [m]</th>
<th>Rot. Error [°]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plane-ISS-SHOT-GCG-PES</td>
<td>2.431</td>
<td>0.036</td>
<td>2.431</td>
</tr>
<tr>
<td>Pass-ISS-SHOT-GCG-PES</td>
<td>2.431</td>
<td>0.036</td>
<td>2.431</td>
</tr>
<tr>
<td>Pass-ISS-SHOT-RANSAC-NCC</td>
<td>7.098</td>
<td>0.108</td>
<td>4.574</td>
</tr>
<tr>
<td>Plane-ISS-SHOT-Hough-PES</td>
<td>9.735</td>
<td>0.147</td>
<td>8.914</td>
</tr>
<tr>
<td>Plane-ISS-SHOT-RANSAC-PES</td>
<td>11.964</td>
<td>0.181</td>
<td>9.972</td>
</tr>
<tr>
<td>Plane-ISS-SHOTC-GCG-PES</td>
<td>2.179</td>
<td>0.020</td>
<td>2.179</td>
</tr>
</tbody>
</table>
4.3 Fine Alignment

In this section we present the results from the experiment on fine alignment convergence. In the plots in figures 4.10 to 4.17 each dot represents a pair of pose errors for the input and output transformations to the fine alignment algorithm. The black vertical line represents the calculated threshold, $\tau$, for failed rough alignment used in section 4.2, at the best determined scale $z_t$.

For each of the datasets, the determined scale seems to give strong predictive power to the initial pose error on the output pose error for small enough input errors. The plots indicate that the fine alignment algorithm has a radius of convergence within which small input errors seem to reasonably predict small output errors. It appears that, after some threshold $\tau$, the predictive power of the input error on the output quickly decreases. This is seen from that the spread of output pose errors increases drastically after some level of initial pose error. Below that level, the spread is virtually invisible when viewed at the scale needed to see the full distribution of results. It further appears that the used method is able to find a reasonable $\tau$, below which the output transformation can be expected to be well behaved. We see this visually, as the vertical point spread is very low to the left of the vertical threshold line, and large to the right of it.

![ICP Analysis: Freiburg1: teddy](image)

*Figure 4.10: ICP Analysis: Freiburg1: teddy*

Each point represents an ICP run on models from the Freiburg1: teddy recording, where the x-axis is the input pose error and the y-axis the output pose error at the best estimated scale $z_t$. The black vertical line represents the detected threshold $\tau$ on the input pose error that will define a failed rough alignment.
4.3 Fine Alignment

**Figure 4.11: ICP Analysis: Freiburg1: xyz**
Each point represents an ICP run on models from the Freiburg1: xyz recording, where the x-axis is the input pose error and the y-axis the output pose error at the best estimated scale $z_t$. The black vertical line represents the detected threshold $\tau$ on the input pose error that will define a failed rough alignment.

**Figure 4.12: ICP Analysis: Freiburg1: plant**
Each point represents an ICP run on models from the Freiburg1: plant recording, where the x-axis is the input pose error and the y-axis the output pose error at the best estimated scale $z_t$. The black vertical line represents the detected threshold $\tau$ on the input pose error that will define a failed rough alignment.
Figure 4.13: ICP Analysis: Freiburg2: coke
Each point represents an ICP run on models from the Freiburg2: coke recording, where the x-axis is the input pose error and the y-axis the output pose error at the best estimated scale $z_t$. The black vertical line represents the detected threshold $\tau$ on the input pose error that will define a failed rough alignment.

Figure 4.14: ICP Analysis: Freiburg2: flowerbouquet
Each point represents an ICP run on models from the Freiburg2: flowerbouquet recording, where the x-axis is the input pose error and the y-axis the output pose error at the best estimated scale $z_t$. The black vertical line represents the detected threshold $\tau$ on the input pose error that will define a failed rough alignment.
4.3 Fine Alignment

**Figure 4.15: ICP Analysis: Freiburg2: flowerbouquet-brownbackground**
Each point represents an ICP run on models from the Freiburg2: flowerbouquet-brownbackground recording, where the x-axis is the input pose error and the y-axis the output pose error at the best estimated scale $z_t$. The black vertical line represents the detected threshold $\tau$ on the input pose error that will define a failed rough alignment.

**Figure 4.16: ICP Analysis: Freiburg3: cabinet**
Each point represents an ICP run on models from the Freiburg3: cabinet recording, where the x-axis is the input pose error and the y-axis the output pose error at the best estimated scale $z_t$. The black vertical line represents the detected threshold $\tau$ on the input pose error that will define a failed rough alignment.
Each point represents an ICP run on models from the Freiburg3: teddy recording, where the x-axis is the input pose error and the y-axis the output pose error at the best estimated scale $z_t$. The black vertical line represents the detected threshold $\tau$ on the input pose error that will define a failed rough alignment.
4.4 Pose Re-Optimization and Fusion

In this section we present the results from the qualitative evaluation of the whole system, focusing mainly on the pose re-optimization and fusion parts. Renderings of the resulting models can be found in appendix B and renderings of the evaluation dataset partial reconstructions dataset in appendix A.2.

For each scene in the partial reconstructions dataset and each pose re-optimization method (or lack thereof), we compare the fused model to the partial reconstructions and to the fused models from the other methods. The results are summarized in tables 4.11 to 4.20. In the tables, DF stands for direct fusion, RTE for re-tracking from estimates, LC for two-map loop-closure, EP for with edge pruning, and CMB for with cross-map building.

We can begin by observing that, with the exception of the foot: clean scene which has a severe deformation in one of the partial reconstructions, alignment succeeded for every scene. However, if we study the direct fusion renderings in appendix B, it is also obvious that none of the registrations of partial reconstructions is perfect, as fusion edges are visible to some degree in every fused model. For example, we can observe the ridge going down the front of the foot model in the second render angle of the scene foot: cluttered (figure B.8). Such edges clearly show that there is an offset in the alignment between the data from the different scans. In addition, we may also observe that there are no resulting models, from any of the methods, that can be considered indistinguishable from that of a single-session reconstruction, as every model has either fusion edges or added reconstruction defects. Added reconstruction defects can be seen for instance in all of the reconstructions of the bust: clean scene (figures B.26 to B.30), where the face looks like it has been squashed from both sides.

In aggregate, the re-track from estimates method has the most good and least bad labels and in general the best looking results. In general, the extended versions of two-map loop-closure have worse looking results than the basic version, although the results are very similar. Overall, the effect of the choice of pose re-optimization strategy is noticeable but not dramatic.

<table>
<thead>
<tr>
<th>Pipeline</th>
<th>Successful Alignment</th>
<th>Reduced Holes</th>
<th>Added Holes</th>
<th>Reduced Defects</th>
<th>Added Defects</th>
<th>Fusion Edges</th>
<th>Main Object Deformations</th>
</tr>
</thead>
<tbody>
<tr>
<td>DF</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>mediocre</td>
<td>mediocre</td>
</tr>
<tr>
<td>RTE</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>good</td>
<td>good</td>
</tr>
<tr>
<td>LC</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>good</td>
<td>good</td>
</tr>
<tr>
<td>LC-EP</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>mediocre</td>
<td>mediocre</td>
</tr>
<tr>
<td>LC-CMB</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>mediocre</td>
<td>mediocre</td>
</tr>
</tbody>
</table>

Table 4.11: Foot: Clean Results Comparison.
Comparison of the results of fusion after different pose re-optimization methods on the Foot: Clean scene.
<table>
<thead>
<tr>
<th>Pipeline</th>
<th>Successful Alignment</th>
<th>Reduced Holes</th>
<th>Added Holes</th>
<th>Reduced Defects</th>
<th>Added Defects</th>
<th>Fusion Edges</th>
<th>Main Object Deformations</th>
</tr>
</thead>
<tbody>
<tr>
<td>DF</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>mediocre</td>
<td>mediocre</td>
</tr>
<tr>
<td>RTE</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>mediocre</td>
<td>mediocre</td>
</tr>
<tr>
<td>LC</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>good</td>
<td>mediocre</td>
</tr>
<tr>
<td>LC-Ep</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>mediocre</td>
<td>mediocre</td>
</tr>
<tr>
<td>LC-CMB</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>bad</td>
<td>bad</td>
</tr>
</tbody>
</table>

**Table 4.12: Foot: Cluttered Results Comparison.**
Comparison of the results of fusion after different pose re-optimization methods on the Foot: Cluttered scene.

<table>
<thead>
<tr>
<th>Pipeline</th>
<th>Successful Alignment</th>
<th>Reduced Holes</th>
<th>Added Holes</th>
<th>Reduced Defects</th>
<th>Added Defects</th>
<th>Fusion Edges</th>
<th>Main Object Deformations</th>
</tr>
</thead>
<tbody>
<tr>
<td>DF</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>mediocre</td>
<td>mediocre</td>
</tr>
<tr>
<td>RTE</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>good</td>
<td>good</td>
</tr>
<tr>
<td>LC</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>mediocre</td>
<td>mediocre</td>
</tr>
<tr>
<td>LC-Ep</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>mediocre</td>
<td>mediocre</td>
</tr>
<tr>
<td>LC-CMB</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>bad</td>
<td>bad</td>
</tr>
</tbody>
</table>

**Table 4.13: Foot: Duplicates Results Comparison.**
Comparison of the results of fusion after different pose re-optimization methods on the Foot: Duplicates scene.

<table>
<thead>
<tr>
<th>Pipeline</th>
<th>Successful Alignment</th>
<th>Reduced Holes</th>
<th>Added Holes</th>
<th>Reduced Defects</th>
<th>Added Defects</th>
<th>Fusion Edges</th>
<th>Main Object Deformations</th>
</tr>
</thead>
<tbody>
<tr>
<td>DF</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>mediocre</td>
<td>mediocre</td>
</tr>
<tr>
<td>RTE</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>good</td>
<td>mediocre</td>
</tr>
<tr>
<td>LC</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>mediocre</td>
<td>mediocre</td>
</tr>
<tr>
<td>LC-Ep</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>bad</td>
<td>bad</td>
</tr>
<tr>
<td>LC-CMB</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>bad</td>
<td>bad</td>
</tr>
</tbody>
</table>

**Table 4.14: Foot: Repositioned Results Comparison.**
Comparison of the results of fusion after different pose re-optimization methods on the Foot: Repositioned scene.

<table>
<thead>
<tr>
<th>Pipeline</th>
<th>Successful Alignment</th>
<th>Reduced Holes</th>
<th>Added Holes</th>
<th>Reduced Defects</th>
<th>Added Defects</th>
<th>Fusion Edges</th>
<th>Main Object Deformations</th>
</tr>
</thead>
<tbody>
<tr>
<td>DF</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>mediocre</td>
<td>mediocre</td>
</tr>
<tr>
<td>RTE</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>good</td>
<td>mediocre</td>
</tr>
<tr>
<td>LC</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>mediocre</td>
<td>mediocre</td>
</tr>
<tr>
<td>LC-Ep</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>bad</td>
<td>bad</td>
</tr>
<tr>
<td>LC-CMB</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>bad</td>
<td>bad</td>
</tr>
</tbody>
</table>

**Table 4.15: Bust: Clean Results Comparison.**
Comparison of the results of fusion after different pose re-optimization methods on the Bust: Clean scene.
### Table 4.16: Bust: Cluttered Results Comparison.
Comparison of the results of fusion after different pose re-optimization methods on the Bust: Cluttered scene.

<table>
<thead>
<tr>
<th>Pipeline</th>
<th>Successful Alignment</th>
<th>Reduced Holes</th>
<th>Added Holes</th>
<th>Reduced Defects</th>
<th>Added Defects</th>
<th>Fusion Edges</th>
<th>Main Object Deformations</th>
</tr>
</thead>
<tbody>
<tr>
<td>DF</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>mediocre</td>
<td>mediocre</td>
</tr>
<tr>
<td>RTE</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>mediocre</td>
<td>mediocre</td>
</tr>
<tr>
<td>LC</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>mediocre</td>
<td>mediocre</td>
</tr>
<tr>
<td>LC-EP</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>mediocre</td>
<td>mediocre</td>
</tr>
<tr>
<td>LC-CMB</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>bad</td>
<td>bad</td>
</tr>
</tbody>
</table>

### Table 4.17: Bust: Duplicates Results Comparison.
Comparison of the results of fusion after different pose re-optimization methods on the Bust: Duplicates scene.

<table>
<thead>
<tr>
<th>Pipeline</th>
<th>Successful Alignment</th>
<th>Reduced Holes</th>
<th>Added Holes</th>
<th>Reduced Defects</th>
<th>Added Defects</th>
<th>Fusion Edges</th>
<th>Main Object Deformations</th>
</tr>
</thead>
<tbody>
<tr>
<td>DF</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>bad</td>
<td>bad</td>
</tr>
<tr>
<td>RTE</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>mediocre</td>
<td>mediocre</td>
</tr>
<tr>
<td>LC</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>mediocre</td>
<td>mediocre</td>
</tr>
<tr>
<td>LC-EP</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>good</td>
<td>mediocre</td>
</tr>
<tr>
<td>LC-CMB</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>bad</td>
<td>bad</td>
</tr>
</tbody>
</table>

### Table 4.18: Bust: Repositioned Results Comparison.
Comparison of the results of fusion after different pose re-optimization methods on the Bust: Repositioned scene.

<table>
<thead>
<tr>
<th>Pipeline</th>
<th>Successful Alignment</th>
<th>Reduced Holes</th>
<th>Added Holes</th>
<th>Reduced Defects</th>
<th>Added Defects</th>
<th>Fusion Edges</th>
<th>Main Object Deformations</th>
</tr>
</thead>
<tbody>
<tr>
<td>DF</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>mediocre</td>
<td>mediocre</td>
</tr>
<tr>
<td>RTE</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>mediocre</td>
<td>good</td>
</tr>
<tr>
<td>LC</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>mediocre</td>
<td>mediocre</td>
</tr>
<tr>
<td>LC-EP</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>mediocre</td>
<td>mediocre</td>
</tr>
<tr>
<td>LC-CMB</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>bad</td>
<td>bad</td>
</tr>
</tbody>
</table>

### Table 4.19: Chair Results Comparison.
Comparison of the results of fusion after different pose re-optimization methods on the Chair scene.
### Table 4.20: Bag Results Comparison.
Comparison of the results of fusion after different pose re-optimization methods on the Bag scene.

<table>
<thead>
<tr>
<th>Pipeline</th>
<th>Successful Alignment</th>
<th>Reduced Holes</th>
<th>Added Holes</th>
<th>Reduced Defects</th>
<th>Added Defects</th>
<th>Fusion Edges</th>
<th>Main Object Deformations</th>
</tr>
</thead>
<tbody>
<tr>
<td>DF</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>mediocre</td>
<td>mediocre</td>
</tr>
<tr>
<td>RTE</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>mediocre</td>
<td>mediocre</td>
</tr>
<tr>
<td>LC</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>mediocre</td>
<td>mediocre</td>
</tr>
<tr>
<td>LC-EP</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>mediocre</td>
<td>mediocre</td>
</tr>
<tr>
<td>LC-CMB</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>mediocre</td>
<td>mediocre</td>
</tr>
</tbody>
</table>
In this chapter we discuss the parts of the results and methodology of this thesis that are particularly interesting or need to be nuanced. We also briefly discuss 3D reconstruction in a larger societal context.

5.1 Results

We begin this section by discussing the results of the rough and fine alignment experiments and end with a discussion on the effects of the different pose re-optimization strategies.

5.1.1 Rough Alignment

Salti et al. [2012] find that the best combination of fixed scale keypoint detector and keypoint descriptor is ISS-SHOT. This detector-descriptor pair is determined to have a good trade-off between accuracy and speed, which is completely consistent with what we find in section 4.1. The FPFH descriptor is constructed by making comparisons both between the keypoint and the points within its support radius, as well as between each point in the keypoint’s support radius, and the points within their support radii. Description speed for FPFH can therefore be expected to degrade quickly for larger support radii compared to that for SHOT, which is constructed through comparisons between only the keypoint and the points within its support. This means that to get reasonable computation times for building FPFH descriptors, one might have to sub-sample the point cloud to match the support radius of the descriptor, which is not done in these experiments. SHOT on the other hand has a higher dimensional descriptor than the FPFH, which means that matching speed should degrade much quicker for larger
sets of SHOT keypoints. Detection with ISS is a good way to decrease the number of keypoints without giving up repeatability, which explains why the ISS-SHOT combination proves so successful. It is not clear however, that the same conclusions would be drawn if the point clouds were also sub-sampled to match the support radius of the used descriptor.

In section 4.2, we see that the less complex registration pipelines, those with RANSAC correspondence clustering, perform worse both in aggregate and with optimized parameters compared to the other evaluated pipelines. We also see that the arguably most complex pipeline, early-integration with plane constrained sparse matching, produces the top performance among all the pipeline configurations. These results motivate a move towards more complex pipelines when using a shape and sparse point matching approach to rough alignment. However, reliably choosing good parameters is hard and requires large and representative datasets to test against. This brings up the question if it might perhaps be more interesting to investigate fine alignment methods with much larger convergence radius than ICP?

The early-integration pipeline with plane constrained sparse matching shows the best alignment performance overall. Although sparse matching is done at a fixed keypoint scale, we can view the method as a coarse to fine optimization in only two steps, where planes are coarse and keypoints fine resolution features. In comparison to the late-integration pipeline, where plane matches are used to generate hypotheses in the same way sparse correspondence groups do, the early-integration pipeline uses the plane matches to constrain the search space of sparse correspondences, which we expect to increase accuracy in all cases where strong plane correspondences exist between scenes. It is not necessarily the case, or perhaps even likely, that planes matched between two reconstructions are both similar and irregular enough that plane alignment can be expected to produce an accurate hypothesis. Therefore, using plane-to-plane generated hypotheses will most likely only improve accuracy in cases where sparse matching fails completely, which is evidenced by the fact that we see in section 4.2 that both Plane-ISS-SHOT-GCG-PES and Pass-ISS-SHOT-GCG-PES produce the same top median pose error.

We see in section 4.2.2 that no pipeline configuration with optimized parameters was able to correctly align the Freiburg2: coke and Freiburg3: cabinet datasets. The coke dataset represents an aluminium coke can on a rectangular table. The aluminium can is highly reflective, which results in poor depth data and subsequently distorted reconstructions. This makes describing repeatable geometric features on the can, in different reconstructions, virtually impossible. The rectangular table, on which the coke can is centered, has a rotational symmetry through 180°. The cabinet, which is placed on the floor and has the shape of a rectangular solid, also has a rotational symmetry through 180°. The cabinet symmetry is broken by the handles on one of the sides but these are not captured or poorly reconstructed in the to reconstruction. We can therefore not reasonably expect an alignment method that only relies on 3D shape to consistently solve these align-
5.1 Results

ment problems. To be able to align several recordings of this type of scene we would most likely need to incorporate some other modality like color.

5.1.2 Fine Alignment

The results from the fine alignment analysis in section 4.3 is interesting in that a clear breakdown point, in terms of the pose error of the initial guess, can be seen for each dataset. We see this in figures 4.10 to 4.17, where the black vertical line depicting the detected threshold $\tau$ marks the breakdown point. It is however very hard to see any other relationship between the input and output pose error than that if the input error is small enough, the output error will be small. Above the breakdown point, the input pose error seems to have very little predictive power on the output error. Ideally, there would be a simple, deterministic, and monotone relationship between the input and output pose error, as this would make it easier to compare rough alignment results. There may exist another practically useful pose error metric that can produce more of such a relationship, but it is also possible that the cost function used for registration contains so many local minima that a more refined pose error metric would be useless.

We see from the results of directly fusing the data from two scans without pose re-optimization in section 4.4 and appendix B, that all alignments are generally correct but not accurate enough to not give artefacts. For seamless fusion of separate recordings, the model-to-model registration must be as accurate as frame registration in sequential tracking, which it in this case is clearly not. Reconstruction artefacts are either due to problems in fusion or tracking and since fusion is handled in the same way for both single- and multi-session reconstruction, multi-session tracking must be as accurate as single-session tracking to achieve equivalent performance. Ideally, pose re-optimization is meant to only have to fix tracking errors in the separate sessions, not fix rigid registration issues between the two pose sets. These results therefore motivate an exploration of other registration methods than ICP for doing fine alignment.

5.1.3 Pose Re-Optimization

We see in section 4.4 and appendix B that all of the proposed methods for pose re-optimization result in models with either fusion edges or additional reconstruction artifacts that were not present in the two separate reconstructed models. Although many of these problems would likely disappear with an accurate enough fine alignment, it is desirable that a pose re-optimization method is somewhat robust to smaller errors in preceding steps.

We observe in particular that the methods based on pose-graph optimization do not improve results as much as expected. The result of a pose-graph optimization is dependent on the measurements $z_{i,j}$ and the information matrices of those measurements $\Omega_{i,j}$, which together make up an edge in the graph. If for some reason an edge is added where $z_{i,j}$ is not actually a measurement of the desired property, in this case a camera pose relative the map, the graph will become a less faithful representation of the true underlying constraints. If for instance the
initial guess for a camera pose places the frame data so far away from the map that all points are considered outliers, $\Omega_{i,j}$ will be undefined. If instead the frame-to-map registration is not allowed to converge, e.g. because of drift, $\Omega_{i,j}$ does not describe the region around the minimum of the cost function.

We expect that pruning edges from matches where the optimization did not converge should improve the result if non-convergence errors are the most important. The results however tell a different story, as edge pruning seems to degrade the results overall. The fact that pruning degrades results hints towards that, whether or not non-convergence errors are the most important, they in some sense balance out some of the effects from registering frames to nearly empty parts of the map. If the map is empty around the frame data after the initial pose guess, the optimization should trivially converge as the cost function is zero. Removing edges from non-converged frames may therefore increase the relative importance of these errors in the graph optimization, which would explain why it degrades performance.

Cross-Map Building is meant to reduce the effects of poorly defined $\Omega_{i,j}$ from trying to register a frame to an empty part of the map. In attempting this, the approach does however break the intended logical structure of the two map-graph by blending the two maps together during registration without representing this explicitly in the graph. The results clearly show that Cross-Map Building as implemented is not an effective strategy as it, like the simple convergence based edge-pruning, seems to degrade the results.

Bylow et al. [2013] report that tracking directly against the TSDF gives more robust tracking with better convergence properties than with the approach taken in KinectFusion [Newcombe et al., 2011], where tracking is done against virtual frames generated from the TSDF. As the distance calculations done for TSDF building is projective, the distances are only accurate near the surface, which motivate the use of a small truncation distance, $\mu$, for accurate sequential tracking. The TSDF is constant further away than $\mu$ from the surface, meaning that points that are guessed to be at least $\mu$ away from the map surface are ignored when optimizing for the camera pose in direct frame-to-TSDF tracking. This means that although this approach gives sequential tracking with low drift, it may not be well suited for cases where surface overlap and accuracy of the initial camera pose guess is lower than for sequential tracking. We expect that to achieve accurate results for multi-session reconstruction, it would need to be paired with a more accurate rigid registration between pose sets, and a more sophisticated method for edge pruning than what we use in this thesis.

In section 4.4, we observe that the best results after fusion is obtained by using the re-tracking from estimates pose re-optimization. The method is basically multi-session reconstruction as continued scanning, where registration is seen as an initialization to tracking the second session. As the information matrix $\Omega_{i,j}$ is not used, we avoid letting bad frame-to-map registrations degrade the whole reconstruction. In particular, we avoid errors due to registering to an empty part of the map without the problems of breaking the logic of a pose-graph. We do this
simply by continuously building the map while using results from single-session tracking as initial guesses. We expect this method to be more robust than pose-graph based methods, but without the potential power to fix larger distortions due to errors in single session tracking.

5.2 Method

In this section we discuss the methodology of this thesis, with a particular focus on the evaluation.

5.2.1 Implementation

The experiments in this thesis regard several variations of a large and quite complex system. The available time therefore necessarily constrains the implementation to rely heavily on using already implemented algorithms, in this case to what is implemented in the PCL library. On the one hand this increases the repeatability of the work as anyone can easily download the source code used in most of the algorithms. On the other hand this means that we may overlook several effective or interesting methods because it would be infeasible to implement them all from scratch in the limited time frame.

5.2.2 Evaluation

The quality of the results, observations, and predictions made in this thesis would be improved by the use of more and better data, as is often the case. We desire a dataset of several RGB-D recordings of the same scene, for many different scenes, with measured ground truth camera trajectories and accurate scene geometry. The scanned scenes should also resemble the intended use cases of a multi-session dense RGB-D SLAM system for 3D reconstruction. In the ideal case there would also be enough recordings to allow for dividing the dataset into a test and an evaluation set, so as to help avoid problems of overfitting parameters. We have yet to find any such datasets after considerable effort.

We choose to use the dataset of RGB-D recordings given in [Sturm et al., 2012], normally used to evaluate SLAM systems, because it includes ground truth camera trajectories. The open source version of KinectFusion, available through PCL, was used together with the ground truth trajectories to simulate a single-session SLAM system in a reasonably repeatable way, in stead of using a potentially more accurate proprietary system. This gives good data for evaluating the registration part of the system. Only a few of the available recordings resemble the desired use case of scanning smaller objects, which results in that the dataset is smaller than we would like.

The rough alignment module has many variations and parameter configurations, all of which would be intractable to analyse in-depth. We therefore choose to focus the analysis on the best performing variations after very broadly evaluating all approaches. There is a risk that methods that could have performed well given better parameter choices are overlooked with this method, but we accept this risk.
by considering that this selection process might also give information on how
easy a method is to tune well.

For evaluation of the whole system, we reason that the TUM dataset is not a good
enough representation of the intended use case. In particular it does not con-
tain any recordings where an object on a table has been repositioned between
scans, which motivates the collection of a new dataset. We choose to only do a
qualitative evaluation of the whole system because it seems unlikely that a quan-
titative one would add much relevant information. The foot used in the partial
reconstruction dataset, exists as a high resolution 3D model and a quantitative
evaluation could consist of looking at the point error distributions between a re-
construction and the ground truth model. Point error analysis carries with it sev-
eral complexities. Most importantly, we cannot know the correct correspondence
between points, and must resort to assigning correspondence based on proximity.
It is not clear how much this approximation affects the results when comparing
models that are so inaccurate that the errors can easily be seen with the naked
eye, which is the case for all resulting models containing the foot. As the results
clearly show that the most important sources of errors in the final results are due
to lacking performance of the fine grained rigid registration, we deem the added
benefit of a quantitative analysis of the pose re-optimization and fusion steps to
be low.

The main purpose of the fine alignment experiment is to determine the appropri-
ate relative weight for comparing rotation and translation errors and gaining a
notion of success for rough alignment. This motivates the focus on predictability
between input and output pose error in the experiment. An alternative approach
to the one used in this thesis is to set absolute bounds on the rotation and trans-
lation errors of the final alignment, and then try to determine bounds on the
input error below which the final bounds can be expected to be achieved. No
conclusive results could be achieved through such a method however. Choosing
fixed bounds for successful fine alignment is both hard and arbitrary and does
not capture the fact that some pairs of models are much easier to register with
ICP than others. As the prediction based analysis is quite successful in describing
the behavior of the fine alignment method, it might be interesting to investigate
further.

5.2.3 Source Criticism

Almost all of the literature that makes up the theoretical basis of this thesis are
articles from peer-reviewed journals and conference papers. The exceptions are
two Master’s theses, two PhD theses, one PhD thesis proposal and the two book
chapters [Bronstein et al., 2012] and [Castellani and Bartoli, 2012]. We consider
these sources to be reliable in general but the main theoretical results are still
reviewed and the central practical results are tested as a part of this thesis. We
critically examine all results and attempt to make use of sources from different
communities to make sure the problem is covered from several perspectives.
5.3 3D Reconstruction In a Larger Context

Cheap and accurate 3D reconstruction together with 3D printing and other automatic manufacturing methods together comprise the ability to digitize physical objects. The ability to digitize sound, written communication and images has already brought with it huge societal consequences. Accompanying increases in speed and reductions in cost of distribution of these media has led to that more people can now consume more art, entertainment and information than could have been imagined before the onset of digitization. The hope is that the digitization of physical objects will bring similar societal benefits in the future. Just like with the digitization of media, it is likely that these developments will make us question the nature of ownership as the line between intellectual and physical ownership continues to blur. It is therefore important that we as a society continue to discuss and debate these issues while we develop the necessary technology.
6.1 Summary

The work in this thesis consists of the implementation and evaluation of several variations on a system for doing multi-session dense RGB-D SLAM for 3D reconstruction. The presented system is designed as an extension to a single-session system with the same purpose, and is mainly focused around handling the relaxation of the incremental camera movement assumption that holds in the single-session case. This is done by first separately reconstructing the object or scene in each session, then bringing the two sessions to the same reference frame and using the data from all sessions to arrive at a final reconstruction. We also investigate a simple case of breaking the static scene assumption to allow an object placed on an empty table to be repositioned between two scanning sessions. We show that the system is able to robustly handle these relaxed assumptions in realistic settings. The goal of this thesis, as presented in section 1.3, is to investigate how best to register and merge two scans, with and without repositioning the objects between scans, which we conclude has been met. The following sections relate to the four questions in the end of the problem statement in section 1.3 according to: registration relates to questions 1 & 2, pose re-optimization and fusion relates to question 3, and repositioned objects relates to question 4.

6.2 Registration

We find that the most important part in extending a single-session 3D reconstruction system to a multi-session system, is the alignment of the session reference frames. We show that roughly aligning the single-session 3D models using an adapted version of a pipeline for object detection can be robust and
accurate enough on scenes without degenerate shape symmetries. We also find that if present, plane correspondences between the two models should be used to constrain the search space in sparse keypoint matching. Simply matching keypoints and using RANSAC to estimate registration parameters between the scenes is shown to not be robust enough. We further show that the variation of ICP recommended in [Rusinkiewicz and Levoy, 2001], is not accurate enough to enable results from multi-session reconstruction to be indistinguishable from single-session reconstructions in terms of quality.

6.3 Pose Re-Optimization and Fusion

The exploration of methods for re-optimizing all camera poses, shows that the straight forward approach of re-tracking the poses sequentially, starting from the aligned results from the single-session reconstructions, gives the best result of the proposed methods. We conclude that pose-graph methods are not well paired with TSDF based frame-to-map tracking using the basic approach that we describe in this thesis. Fusion of the data from several sessions can easily be done by building a single TSDF using the frames from all sessions once the camera poses have been estimated in the same reference frame.

6.4 Repositioned Objects

To reconstruct objects that have been repositioned between sessions, we detect and remove the background after single-session reconstruction and then treat it as a regular multi-session problem. Repositioning an object between scans necessarily decreases the amount of overlap between scans, which makes accurate registration harder and calls for a fine alignment algorithm that can handle lower degrees of overlap than ICP is able to.

6.5 Future Work

In this thesis we have mainly considered fixed scale keypoint detection for speed and complexity reasons. However, interesting features no doubt exist at many scales and making use of this might very well increase accuracy and robustness, as hinted by the success of scale space detectors in 2D images. This could either be done by employing a fixed scale detector at multiple scales, using an adaptive scale detector, or trying to match more large scale features like planes. As evidenced by the failure to consistently align highly geometrically symmetric objects it would likely be beneficial to incorporate color information in either feature description or hypothesis verification.

To be able to produce as accurate and good looking results with multi-session as with single-session reconstruction, it is necessary to explore other methods for fine alignment than ICP. As point clouds from separate scanning sessions often
have far from full overlap, it would be interesting to evaluate methods that use more robust error measures than point-to-plane $L_2$ distance, such as for instance LM-ICP [Fitzgibbon, 2003] or Sparse ICP [Bouaziz et al., 2013].

As errors can occur for single-session reconstruction in many real-world situations, it is desirable that a multi-session system reduces as much of them as possible. More powerful pose re-optimization strategies are therefore needed and it would be interesting to continue to investigate pose-graph optimization based methods. One approach to improve the two-map pose-graph methods presented in this thesis would be to use a distance transform on the single-session models to create distance fields to track against, instead of tracking against an incrementally built a TSDF. In combination with more intelligent edge-pruning this might very well fix the problems discussed in section 5.1.3.
Appendix
In this appendix we show renderings of the datasets used to evaluate the built system in this thesis.

A.1 TUM RGB-D Dataset

This section contains renderings of the *TUM RGB-D Dataset*. Each used RGB-D recording has been divided in two equally long parts, referred to here as *From* and *To*, which have been used to reconstruct two models of the recorded scene.
<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
</tr>
</thead>
</table>

**Figure A.1: Freiburg1: teddy.**
Renderings of the From (left) and To (right) reconstructions from the Freiburg1: teddy recording.
Renderings of the From (left) and To (right) reconstructions from the Freiburg1: xyz recording.

*Figure A.2: Freiburg1: xyz.*
Figure A.3: Freiburg1: plant.
Renderings of the From (left) and To (right) reconstructions from the Freiburg1: plant recording.
From | To
--- | ---

**Figure A.4:** Freiburg2: coke.
Renderings of the From (left) and To (right) reconstructions from the Freiburg2: coke recording.
Figure A.5: Freiburg2: flowerbouquet.
Renderings of the From (left) and To (right) reconstructions from the Freiburg2: flowerbouquet recording.
Figure A.6: Freiburg2: flowerbouquet-brownbackground. Renderings of the From (left) and To (right) reconstructions from the Freiburg2: flowerbouquet-brownbackground recording.
Figure A.7: Freiburg3: cabinet.
Renderings of the From (left) and To (right) reconstructions from the Freiburg3: cabinet recording.
Figure A.8: Freiburg3: teddy.
Renderings of the From (left) and To (right) reconstructions from the Freiburg3: teddy recording.
A.2 Partial Reconstruction Dataset

This section contains renderings of the Partial Reconstruction Dataset. Each scene has been captured in two RGB-D video recordings, referred to here as From and To, which have been used to reconstruct two models of the recorded scene.

Figure A.9: Foot:Clean. Renderings of the From (left) and To (right) reconstructions of the Foot:Clean scene.
Figure A.10: Bust:Clean.
Renderings of the From (left) and To (right) reconstructions of the Bust:Clean scene.
Figure A.11: Foot:Cluttered.
Renderings of the From (left) and To (right) reconstructions of the Foot:Cluttered scene.
Figure A.12: Bust:Cluttered.
Renderings of the From (left) and To (right) reconstructions of the Bust:Cluttered scene.
Figure A.13: Foot:Duplicates.
Renderings of the From (left) and To (right) reconstructions of the Foot:Duplicates scene.
Figure A.14: Bust:Duplicates.
Renderings of the From (left) and To (right) reconstructions of the Bust:Duplicates scene.
Figure A.15: Foot:Repositioned.
Renderings of the From (left) and To (right) reconstructions of the Foot:Repositioned scene.
Figure A.16: Bust: Repositioned. Renderings of the From (left) and To (right) reconstructions of the Bust: Repositioned scene.
### Figure A.17: Chair.
Renderings of the From (left) and To (right) reconstructions of the Chair scene.
**Figure A.18: Bag.**
Renderings of the From (left) and To (right) reconstructions of the Bag scene.
In this appendix we show renderings of the results to the whole system evaluation on the Partial Reconstruction Dataset.

### B.1 Foot: Clean

(a) First example frame.  
(b) Second example frame.

**Figure B.1: Example Frames**  
Example frames from the Foot: Clean recording.
Figure B.2: Direct Fusion
Result of fusing Foot: Clean using the Direct Fusion approach.

Figure B.3: Re-Tracking From Estimates
Result of fusing Foot: Clean using the Re-Tracking From Estimates approach.

Figure B.4: Two-Map Loop-Closure
Result of fusing Foot: Clean using the Two-Map Loop-Closure approach.
Figure B.5: Two-Map Loop-Closure With Edge-Pruning
Result of fusing Foot: Clean using the Two-Map Loop-Closure With Edge-Pruning approach.

Figure B.6: Two-Map Loop-Closure With Cross-Map Building
Result of fusing Foot: Clean using the Two-Map Loop-Closure With Cross-Map Building approach.
B.2 Foot: Cluttered

(a) First example frame.  
(b) Second example frame.

Figure B.7: Example Frames
Example frames from the Foot: Cluttered recording.

(a) First render angle.  
(b) Second render angle.

Figure B.8: Direct Fusion
Result of fusing Foot: Cluttered using the Direct Fusion approach.
(a) First render angle.  
(b) Second render angle.

**Figure B.9: Re-Tracking From Estimates**
Result of fusing Foot: Cluttered using the Re-Tracking From Estimates approach.

(a) First render angle.  
(b) Second render angle.

**Figure B.10: Two-Map Loop-Closure**
Result of fusing Foot: Cluttered using the Two-Map Loop-Closure approach.

(a) First render angle.  
(b) Second render angle.

**Figure B.11: Two-Map Loop-Closure With Edge-Pruning**
Result of fusing Foot: Cluttered using the Two-Map Loop-Closure With Edge-Pruning approach.
Figure B.12: Two-Map Loop-Closure With Cross-Map Building
Result of fusing Foot: Cluttered using the Two-Map Loop-Closure With Cross-Map Building approach.
B.3 Foot: Duplicates

(a) First example frame.  
(b) Second example frame.

**Figure B.13: Example Frames**
Example frames from the Foot: Duplicates recording.

(a) First render angle.  
(b) Second render angle.

**Figure B.14: Direct Fusion**
Result of fusing Foot: Duplicates using the Direct Fusion approach.
Figure B.15: Re-Tracking From Estimates
Result of fusing Foot: Duplicates using the Re-Tracking From Estimates approach.

Figure B.16: Two-Map Loop-Closure
Result of fusing Foot: Duplicates using the Two-Map Loop-Closure approach.

Figure B.17: Two-Map Loop-Closure With Edge-Pruning
Result of fusing Foot: Duplicates using the Two-Map Loop-Closure With Edge-Pruning approach.
Figure B.18: Two-Map Loop-Closure With Cross-Map Building
Result of fusing Foot: Duplicates using the Two-Map Loop-Closure With Cross-Map Building approach.

(a) First render angle.  
(b) Second render angle.
B.4 Foot: Repositioned

Figure B.19: Example Frames
Example frames from the Foot: Repositioned recording.

Figure B.20: Direct Fusion
Result of fusing Foot: Repositioned using the Direct Fusion approach.
B.4 Foot: Repositioned

Figure B.21: Re-Tracking From Estimates
Result of fusing Foot: Repositioned using the Re-Tracking From Estimates approach.

Figure B.22: Two-Map Loop-Closure
Result of fusing Foot: Repositioned using the Two-Map Loop-Closure approach.

Figure B.23: Two-Map Loop-Closure With Edge-Pruning
Result of fusing Foot: Repositioned using the Two-Map Loop-Closure With Edge-Pruning approach.
Figure B.24: Two-Map Loop-Closure With Cross-Map Building
Result of fusing Foot: Repositioned using the Two-Map Loop-Closure With Cross-Map Building approach.
B.5  Bust: Clean

Figure B.25: Example Frames
Example frames from the Bust: Clean recording.

Figure B.26: Direct Fusion
Result of fusing Bust: Clean using the Direct Fusion approach.
Figure B.27: Re-Tracking From Estimates
Result of fusing Bust: Clean using the Re-Tracking From Estimates approach.

Figure B.28: Two-Map Loop-Closure
Result of fusing Bust: Clean using the Two-Map Loop-Closure approach.

Figure B.29: Two-Map Loop-Closure With Edge-Pruning
Result of fusing Bust: Clean using the Two-Map Loop-Closure With Edge-Pruning approach.
B.5  Bust: Clean

Figure B.30: Two-Map Loop-Closure With Cross-Map Building
Result of fusing Bust: Clean using the Two-Map Loop-Closure With Cross-Map Building approach.
B.6  Bust: Cluttered

Figure B.31: Example Frames
Example frames from the Bust: Cluttered recording.

Figure B.32: Direct Fusion
Result of fusing Bust: Cluttered using the Direct Fusion approach.
Figure B.33: Re-Tracking From Estimates
Result of fusing Foot: Cluttered using the Re-Tracking From Estimates approach.

Figure B.34: Two-Map Loop-Closure
Result of fusing Bust: Cluttered using the Two-Map Loop-Closure approach.

Figure B.35: Two-Map Loop-Closure With Edge-Pruning
Result of fusing Bust: Cluttered using the Two-Map Loop-Closure With Edge-Pruning approach.
Figure B.36: Two-Map Loop-Closure With Cross-Map Building
Result of fusing Bust: Cluttered using the Two-Map Loop-Closure With Cross-Map Building approach.
B.7 Bust: Duplicates

(a) First example frame.  
(b) Second example frame.

Figure B.37: Example Frames  
Example frames from the Bust: Duplicates recording.

(a) First render angle.  
(b) Second render angle.

Figure B.38: Direct Fusion  
Result of fusing Bust: Duplicates using the Direct Fusion approach.
Figure B.39: Re-Tracking From Estimates
Result of fusing Bust: Duplicates using the Re-Tracking From Estimates approach.

Figure B.40: Two-Map Loop-Closure
Result of fusing Bust: Duplicates using the Two-Map Loop-Closure approach.

Figure B.41: Two-Map Loop-Closure With Edge-Pruning
Result of fusing Bust: Duplicates using the Two-Map Loop-Closure With Edge-Pruning approach.
Figure B.42: Two-Map Loop-Closure With Cross-Map Building
Result of fusing Bust: Duplicates using the Two-Map Loop-Closure With Cross-Map Building approach.
B.8 Bust: Repositioned

(a) First example frame.  
(b) Second example frame.

*Figure B.43: Example Frames*  
Example frames from the Bust: Repositioned recording.

(a) First render angle.  
(b) Second render angle.

*Figure B.44: Direct Fusion*  
Result of fusing Bust: Repositioned using the Direct Fusion approach.
Figure B.45: Re-Tracking From Estimates
Result of fusing Bust: Repositioned using the Re-Tracking From Estimates approach.

Figure B.46: Two-Map Loop-Closure
Result of fusing Bust: Repositioned using the Two-Map Loop-Closure approach.

Figure B.47: Two-Map Loop-Closure With Edge-Pruning
Result of fusing Bust: Repositioned using the Two-Map Loop-Closure With Edge-Pruning approach.
Figure B.48: Two-Map Loop-Closure With Cross-Map Building
Result of fusing Bust: Repositioned using the Two-Map Loop-Closure With Cross-Map Building approach.
B.9 Chair

(a) First example frame.  
(b) Second example frame.

Figure B.49: Example Frames
Example frames from the Chair recording.

(a) First render angle.  
(b) Second render angle.

Figure B.50: Direct Fusion
Result of fusing Chair using the Direct Fusion approach.
Figure B.51: Re-Tracking From Estimates
Result of fusing Chair using the Re-Tracking From Estimates approach.

(a) First render angle.  
(b) Second render angle.

Figure B.52: Two-Map Loop-Closure
Result of fusing Chair using the Two-Map Loop-Closure approach.

(a) First render angle.  
(b) Second render angle.

Figure B.53: Two-Map Loop-Closure With Edge-Pruning
Result of fusing Chair using the Two-Map Loop-Closure With Edge-Pruning approach.
B.9 Chair

Figure B.54: Two-Map Loop-Closure With Cross-Map Building
Result of fusing Chair using the Two-Map Loop-Closure With Cross-Map Building approach.
B.10 Bag

Figure B.55: Example Frames
Example frames from the Bag recording.

Figure B.56: Direct Fusion
Result of fusing Bag using the Direct Fusion approach.
Figure B.57: Re-Tracking From Estimates
Result of fusing Bag using the Re-Tracking From Estimates approach.

Figure B.58: Two-Map Loop-Closure
Result of fusing Bag using the Two-Map Loop-Closure approach.

Figure B.59: Two-Map Loop-Closure With Edge-Pruning
Result of fusing Bag using the Two-Map Loop-Closure With Edge-Pruning approach.
Figure B.60: Two-Map Loop-Closure With Cross-Map Building
Result of fusing Bag using the Two-Map Loop-Closure With Cross-Map Building approach.
Bibliography


Umberto Castellani and Adrien Bartoli. 3d shape registration. In Nick Pears, Yonghuai Liu, and Peter Bunting, editors, 3D Imaging, Analysis and Appli-


